PROYECTO FINAL DE CARRERA

TÍTULO: Medidor de Distancias por Ultrasonidos

AUTOR: Oliver Sánchez Blanco

TITULACIÓN: E.T.T. Sistemas Electrónicos

DIRECTOR: Ramón Guzmán Solà

DEPARTAMENTO: Teoría de la Señal y Comunicaciones

FECHA: 6 de Julio de 2011

TÍTULO: Medidor de Distancias por Ultrasonidos

APELLIDOS: Sánchez Blanco **NOMBRE: Oliver**

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones

ESPECIALIDAD: Sistemas Electrónicos **PLAN: 95**

DIRECTOR: Ramón Guzmán Solà

DEPARTAMENTO: Teoría de la Señal y Comunicaciones

QUALIFICACIÓN DEL PFC

TRIBUNAL

PRESIDENTE	SECRETARIO	VOCAL
Joan Vicent Castell Balaguer	Vicent Sales Zaragozà	Antonio López Martínez

FECHA DE LECTURA: 6 de Julio de 2011

Este Proyecto tiene en cuenta aspectos medioambientales: ■ Sí □ No

PROYECTO FINAL DE CARRERA

RESUMEN (máximo 50 líneas)

El proyecto consiste en la realización de un medidor de distancias, para ello se ha utilizado un sensor de ultrasonidos y un microcontrolador.

El sensor nos proporciona la señal que necesitamos para medir la distancia y con el microcontrolador trabajamos dicha señal para poder obtener los datos que nos interesa y poder visualizarlos a través de un módulo LCD.

Este proyecto consta de tres fases:

- Diseño del prototipo por software y posterior simulación.
- Verificación del diseño mediante hardware.
- Fabricación del prototipo final.

Para el diseño se han utilizado los siguientes programas:

- MPLAB para la programación del microcontrolador PIC16F84A, utilizando el lenguaje de programación Ensamblador (Assembler).
- PROTEUS para la simulación y posterior fabricación.

Palabras clave (máximo 10):

Sensor	Ultrasonidos	Microcontrolador	PIC16F84A
MPLAB	Ensamblador	PROTEUS	

Propuesta

EPSEVG - ESCOLA POLITÈCNICA SUPERIOR D'ENGINYERIA... http://www2.epsevg.upc.edu/campusdigital/propostesPFC_print3.asp

2010.11.2

Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROPOSTA DE PROJECTE FINAL DE CARRERA Pàg: 1/2

OLIVER SANCHEZ BLANCO -DNI: 44196617 - Telèfon: 933370445
ENGINYERIA TÈCNICA DE TELECOMUNICACIÓ, ESPECIALITAT EN SISTEMES ELECTRÒNICS - 340ETSE 95

17/3/2011
Signatura

Projecte proposat per:

☒ **1 Departament:**
739, TEORIA DEL SENYAL I COMUNICACIONS

Projecte

- **Títol del projecte**
  Medidor de Distancias por Ultrasonidos

- **Estudiant/a**
  OLIVER SANCHEZ BLANCO

- **Director/a del projecte**
  Ramón Guzmán Solà

Signatura del director

- **Professor/a Ponent (en el cas de projectes realitzats en una empresa)**

Signatura del ponent

- **Vist i Plau Cap de secció departament (per a tots els tipus de propostes)**
  Departament: 739, TEORIA DEL SENYAL I COMUNICACIONS
  Cap de seccio: ANTONI BARLABE DALMAU

Signatura del cap de secció

DATA I SIGNATURA APROVACIÓ COMISSIÓ COORDINACIÓ DOCENT

Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Copia per l'Estudiant

1 de 2 17/03/2011 17:19

Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROPOSTA DE PROJECTE FINAL DE CARRERA

OLIVER SANCHEZ BLANCO -DNI: 44196617 - Telèfon: 933370445
ENGINYERIA TÈCNICA DE TELECOMUNICACIÓ, ESPECIALITAT EN SISTEMES ELECTRÒNICS - 340ETSE 95

Objectius / Programació / Recursos

Objectius a assolir
Realizar un dispositivo capaz de medir distancias mediante un sensor ultrasonidos y un PIC.

Descripció i Programació temporal del treball a realitzar
1- Recopilación de información y comparativa entre los diferentes sensores y la elección final.

2 - Estudiar el funcionamiento del PIC.

3 - Estudiar las herramientas de programación del PIC y su lenguaje de programación (Assembler).

4 - Estudiar el programa Proteus para la realización de la simulación del Micro .

5 - Realización de placa de pruebas para la verificación en modo real de los elementos que componen el circuito (PIC, Sensor, etc.)

6 - Toma de medidas y calibración

7 - Realización de layout y montaje para presentación final.

Recursos del Centre
Laboratorio de Teoria de la Señal y Comunicaciones

LGEU

Nom dels estudiants que realitzen el projecte

OLIVER SANCHEZ BLANCO
OLIVER SANCHEZ BLANCO

Agradecimientos

Mi principal agradecimiento es para mi familia, en especial a mi padre y a mi madre por su gran apoyo durante la realización de este proyecto final de carrera, por todos mis años de estudio y por su gran confianza puesta en mí.

Agradecer a mis amigos su gran apoyo y ánimo durante todo el tiempo que he estado realizando este proyecto final de carrera.

Agradecer a mi tutor, Ramón Guzmán, las horas dedicadas en mí durante todo el tiempo que ha durado el diseño de este proyecto.

Agradecer a los profesores Joan Vicent Catell y Vicent Sales por ofrecerme su ayuda y sus consejos a la hora de superar inconvenientes encontrados durante la realización de este proyecto.

Agradecer a mis compañeros del LGEU y en especial a Oscar De Sousa por sus aportaciones en materia de electrónica y las facilidades otorgadas a la hora de necesitar cualquier ayuda.

Y un especial agradecimiento al señor Albert Dols por sus consejos y su gran ayuda a la hora de realizar la estructura final del prototipo.

A todos ellos muchas gracias, Oliver

Índice de Contenidos

Objetivos

Objetivos a alcanzar

Realizar un dispositivo capaz de medir distancias mediante un sensor de ultrasonidos y un PIC.

Descripción y Programación temporal del trabajo a realizar

1. Recopilación de información y comparativa entre los diferentes sensores y elección final.
2. Estudiar el funcionamiento del PIC.
3. Estudiar las herramientas de programación del PIC y su lenguaje de programación (Assembler).
4. Estudiar el programa Proteus para la realización de la simulación del micro.
5. Realización de placa de pruebas para la verificación en modo real de los elementos que componen el circuito (PIC, Sensor, etc.).
6. Toma de medidas y calibración.
7. Realización del layout y montaje para presentación final.

Recursos del Centro

- Laboratorio de Teoría de la Señal y Comunicaciones.
- LGEU

Introducción del Proyecto

El proyecto consiste en la realización de un medidor de distancias por ultrasonidos. En cuya realización ha sido utilizado un sensor de ultrasonidos y un Microcontrolador (PIC16F84A), como dispositivos principales para la realización del mismo.

Para la realización del proyecto se han seguido los siguientes pasos:

1. Diseño del prototipo por software y posterior simulación.
2. Verificación del diseño mediante hardware.
3. Fabricación del prototipo final.

En el diseño del prototipo se han utilizado los siguientes programas de desarrollo:

- MPLAB, software que nos permite la programación del Microcontrolador. Dicho programa es proporcionado por el fabricante del PIC (*Microchip*). El lenguaje de programación utilizado es Ensamblador (*Assembler*).
- PROTEUS, software que nos permite realizar la simulación del PIC y su posterior verificación de código fuente. Para el proyecto se utiliza la vertiente ISIS para el desarrollo de la simulación y el diseño del esquema del circuito, y la vertiente ARES para la fabricación del PCB (*Printed Circuit Board*).

Para la posterior verificación del prototipo diseñado se construye una placa de pruebas, donde podemos comprobar las mediciones en un entorno físico y realizar las posibles calibraciones o mejoras. Todo ello visualizado a través de un módulo LCD (*Liquid Crystal Display*).

Por último se pasa a construir el prototipo para la presentación final.

Capítulo 1. Estudio del Sensor

1.1 Definición de sensor

Un sensor está compuesto de los siguientes elementos:

Transductor: Dispositivo que transforma una magnitud física de entrada (luz, sonido, temperatura, etc.) en otra de salida (por norma general suelen ser valores de tensión).

Procesador de señal: Dispositivo que realiza una cierta operación con una señal, como por ejemplo una amplificación, un filtrado, etc.

Esto sería la descripción bajo un punto de vista de bloques. De forma funcional podemos decir que un sensor es un dispositivo que capta la magnitud de una variable física en un sistema físico o entorno. Cualquier dispositivo que es alterado por las variaciones de una magnitud física de una forma predecible y medible es un sensor para esa magnitud. Así, un sensor viene caracterizado por su función de transferencia, que relaciona el valor de la magnitud física con el valor que el sensor suministra en su salida.

1.2 Características que definen un sensor

- **Accesibilidad**. Es la zona del entorno físico en la que las variaciones de la magnitud a medir afectan al sensor.
- **Rango de operación**. Los sensores operan sólo en un determinado rango de valores de la magnitud a medir e incluso de otras magnitudes.
- **Datos**. Formato de los datos. Logarítmicos o lineales, discretos o continuos, procesamiento local, ancho de banda, capacidades de compresión de la información, etc.
- **Sensibilidad**. Especificaciones sobre exactitud y precisión.
- **Localización**. Local o remoto al lugar de procesamiento.
- **Inteligencia**. Se dice que un sensor es inteligente si tiene capacidades de procesamiento o decisión. El uso de sensores inteligentes permite establecer un compromiso entre computación y comunicación. La capacidad de procesamiento local pueden reducir las necesidades de ancho de banda para una red de sensores.

1.3 Elección del sensor

Para la realización de un medidor de distancias, se pueden utilizar varias tecnologías. Es posible utilizar sensores de ultrasonidos, infrarrojos, láser o de radiofrecuencia.

1. **Sensores de ultrasonidos:** Son sensores útiles para medir distancias comprendidas entre un rango de 4cm hasta 3-5 metros. Su tecnología hace que obtengamos una señal bastante fiable y manejable de procesar ya que es digital.
2. **Sensores Infrarrojos:** Son utilizados para medir distancias con más precisión debido a que su haz es más estrecho y no tan cónico como los ultrasonidos. Las señales que se obtienen con estos sensores mayoritariamente son analógicas y con comportamiento logarítmico. Haciendo necesario utilizar dispositivos adicionales para poder trabajar digitalmente. No obstante también existen sensores infrarrojos con respuesta digital pero su precio es considerablemente más elevado.
3. **Sensores láser:** El comportamiento es similar al de infrarrojos pero con la ventaja de que tiene un alcance mayor, por lo tanto es ideal para medir distancias largas. Su gran inconveniencia es el coste de estos sensores, siendo bastantes más caros que los anteriores.
4. **Radiofrecuencia:** Son sistemas utilizados para medir distancias de alto rango, por lo tanto hace falta una circuitería que trabaje a alta frecuencia. Sistemas de alta potencia para grandes distancias, y detección de blancos grandes, sistemas RADAR (*Radio Detecting and Ranging*) por lo tanto no factible para el propósito.

Para desarrollar un proyecto de este tipo hay que tener en cuenta lo siguiente:

- **Tamaño:** Es conveniente tener presente el tamaño del montaje debido a que la finalidad del proyecto es hacerlo portátil y manejable.
- **Precio:** Al ser un proyecto final de carrera es adecuado elegir un sensor que tenga una buena relación calidad/precio, ya que no se dispone de un gran presupuesto.
- **Complejidad en el procesado de la Información:** Debido a que se utiliza un PIC para el cálculo de la distancia, se tiene que tener en cuenta la información que nos aporta el sensor. Intentando encontrar un tipo de señal lo más manejable posible a la hora de tratarla. Por ejemplo, una buena señal sería un pulso digital.
- **Complejidad en el diseño:** Esta necesidad depende de la anterior, si la señal que nos proporciona el sensor es buena y fácil de procesar, será más fácil hacer el diseño del circuito. Ya que nos ahorramos dispositivos digitales, como por ejemplo un convertidor A/D.

La finalidad es diseñar un medidor de distancias, a modo didáctico, a objetos fijos y de distancias relativamente cortas. Teniendo en cuenta los apartados explicados anteriormente, se ha optado por utilizar un sensor de ultrasonidos.

1.4 Estudio de los sensores de ultrasonidos

A continuación se definen algunos conceptos que hacen referencia a los sensores de este tipo:

1.4.1 Zona Muerta

Los sensores de ultrasonidos tienen una zona muerta en la cual no pueden detectar exactamente el objeto u obstáculo. Esta es la distancia entre la membrana de sensor y el mínimo rango de sensibilidad. Si el objeto está demasiado cercano, la señal ultrasónica puede chocar contra el objeto antes de que dicha señal haya dejado el transductor, por lo tanto, la información del eco devuelta al sensor es ignorada por el transductor, puesto que éste todavía está transmitiendo y no recibiendo. También puede ocurrir otro problema, que el eco generado se refleje sobre la membrana del sensor y viaje de nuevo hacia el objeto. Estos ecos múltiples pueden dar lugar a errores cuando el objeto está dentro de la zona muerta.

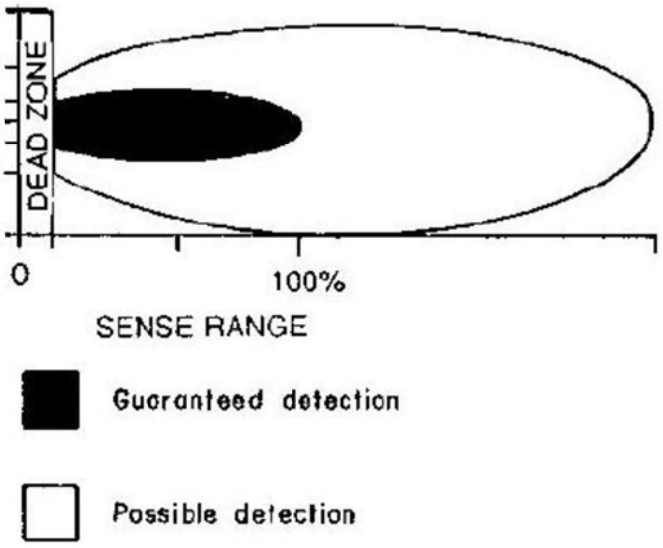


Figura 1.1 *Zona Muerta*

1.4.2 Máximo Rango Sensible

El rango máximo en el que se puede detectar cada objeto y cada aplicación se determina mediante experimentación. En las figuras siguientes, se muestran las características de sensibilidad y las distancias sensibles típicas para el sensor de ultrasonidos.

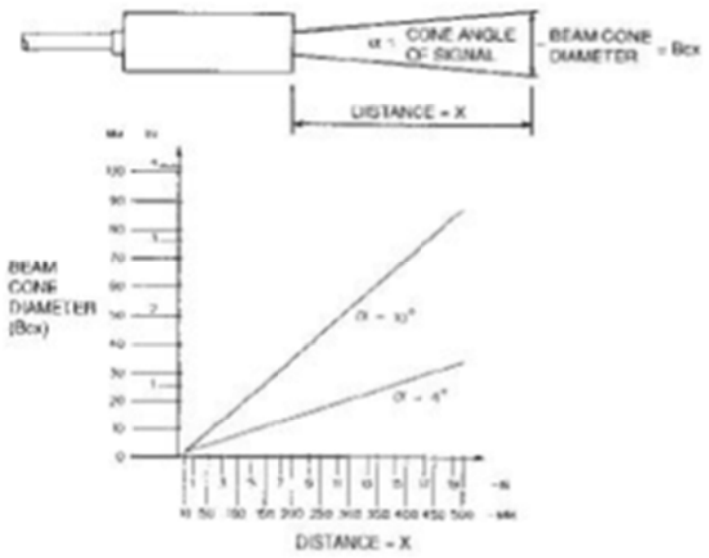


Figura 1.2 *Cono de emisión*

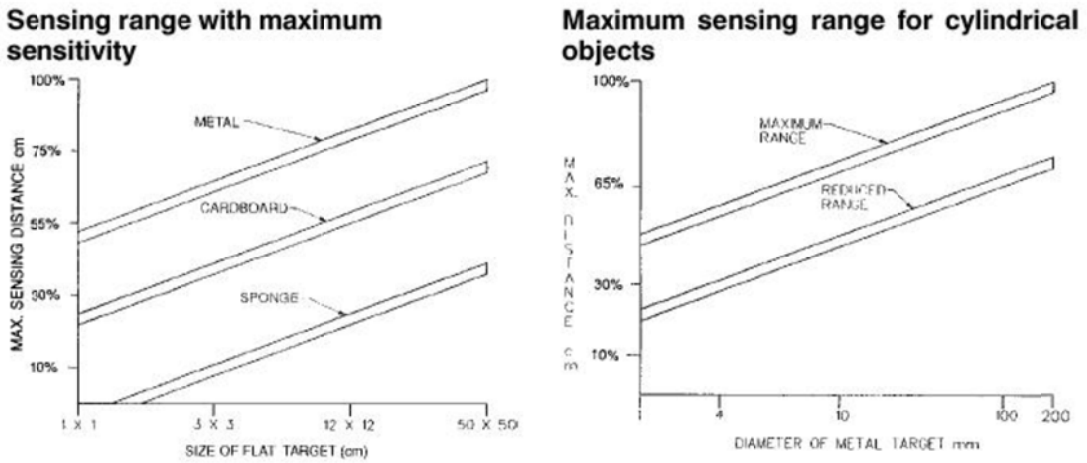


Figura 1.3 *Comparaciones*

1.4.3 Ángulo de Emisión

El ángulo del cono de emisión está formado por los puntos del espacio en los que la señal del sensor es atenuada por lo menos 3 dB. Fuera de este cono, la señal de ultrasonidos existe pero es bastante débil. Este cono debe determinarse experimentalmente y dentro de él pueden detectarse los objetos.

1.4.4 Diámetro del Cono de Emisión

El sensor de ultrasonidos emite un haz de sonido en forma de cono que elimina los lóbulos laterales. Es importante el tamaño del objeto respecto del tamaño de la zona que abarca el haz. Teóricamente, el objeto más pequeño detectable es aquel que mide la mitad de la longitud de onda de la señal del sensor de ultrasonidos. Normalmente los objetos suelen ser grandes, por lo que son detectados a varias distancias. Para determinar el área que abarca el sensor de ultrasonidos a una determinada distancia (diámetro del cono de emisión), se usa la siguiente fórmula:

$$BOX - 2 \cdot X \cdot \tan\left(\frac{\propto}{2}\right) \qquad (1.1)$$

Donde BOX es el diámetro del cono de emisión a la distancia X.
X es la distancia del objeto (obstáculo) al sensor.
α es el ángulo del cono de emisión.

1.4.5 Frecuencia de disparo

La máxima frecuencia a la que un sensor es capaz de dispararse o pararse depende de varias variables, las más significativas son:

- El tamaño del objeto
- El material del que está hecho
- La distancia a la que se encuentra

De este modo, la máxima frecuencia para un objeto pequeño será menor que para un objeto grande. Los materiales que absorben el sonido de altas frecuencias (algodón, esponja, etc.) son más difíciles de detectar que el acero, el cristal o el plástico. De este modo, ellos tienen también una menor máxima de frecuencia cambiante.

La distancia del objeto al sensor es muy importante para determinar el máximo de frecuencia de disparo. El sensor manda una señal ultrasónica por el aire; la señal deja el sensor; viaja hasta el objeto; choca contra él y vuelve hasta el sensor como un eco.

1.4.6 Inclinación del Haz de Ultrasonidos

Si un objeto liso es inclinado más de ± 3 grados con respecto a la normal al eje del haz de emisión de la señal de ultrasonidos, parte de la señal es desviada del sensor y la distancia de detección disminuye. Sin embargo, para objetos pequeños situados cerca del sensor, la desviación respecto a la normal puede aumentar hasta ± 8 grados. Si el objeto está inclinado más de ± 12 grados respecto a la normal, toda la señal es desviada fuera del sensor y el sensor no responderá. La señal que choca contra un objeto de superficie rugosa (como por ejemplo un material granulado), se difunde y refleja en todas las direcciones y parte de la energía vuelve al sensor como un eco débil.

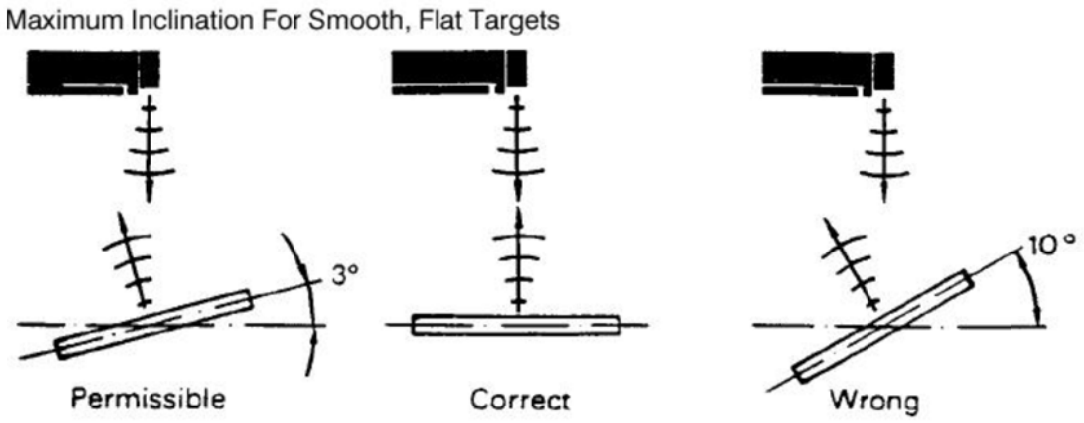


Figura 1.4 *Dependencia del rango de inclinación*

1.5 Características Físicas de la Onda Ultrasónica

Las ondas de ultrasonido se han utilizado para la determinación de la distancia, así mismo se han empleado para la ubicación de un objeto en el espacio, sistemas de construcción, robótica, etc.

El funcionamiento básico del sensor de ultrasonidos se basa en la medida del tiempo transcurrido entre la emisión de un ultrasonido y la recepción del eco correspondiente al mismo. Conocido el tiempo, y siguiendo la velocidad de propagación del sonido en el aire, que viene indicada mediante la siguiente expresión:

$$v = 331,6 + 0,6 \cdot T \qquad (1.2)$$

Donde la *T* (Temperatura) viene dada en grados centígrados y la v (velocidad) en m/s. La distancia a la que se encuentra el objeto que ha devuelto el eco se calcula según la siguiente ecuación:

$$\boldsymbol{d = (331,6 + 0,6 \cdot T) \cdot t/2} \qquad \textbf{\textit{(1.3)}}$$

Donde *t* representa el tiempo transcurrido entre la emisión y la recepción.

El tiempo es dividido por 2 para calcular sólo el tiempo que tarda en llegar la onda al objeto. El empleo de ondas de ultrasonidos en lugar de ondas electromagnéticas se justifica por los puntos siguientes:

- Las ondas acústicas, al contrario que las electromagnéticas, requieren de un medio para transmitirse; como puede ser el aire.
- La velocidad de transmisión de las ondas ultrasónicas es mucho menor que la de las electromagnéticas (velocidad de la luz). Esta característica permite emplearlas para la medida de distancias pequeñas.
- La longitud de onda de un ultrasonido permite que la reflexión que se produce en la mayoría de los objetos sea especular y no difusa. Debido a que la medida proporcionada por la longitud de onda es mayor que la rugosidad de la mayoría de las superficies.

La apertura del cono de emisión constituye en realidad una aproximación del lóbulo central de emisión, pues la expresión de la onda emitida es en realidad más compleja, siguiendo un patrón como el mostrado en la figura [1.5].

1.6 Estudio de las Restricciones de Percepción: Sensores de Ultrasonidos

Los sensores de ultrasonidos no son ideales, es decir, producen lecturas bastante aproximadas a la realidad.

A continuación se muestra un mapa del comportamiento de la energía emitida por la onda ultrasónica y el rango efectivo del sensor utilizado en este proyecto (SRF04):

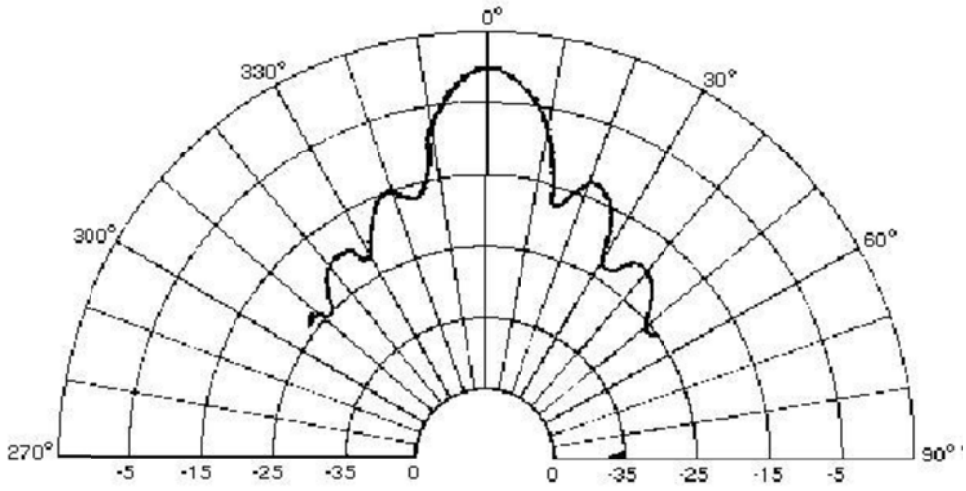


Figura 1.5 *Energía emitida por la onda ultrasónica en todas las direcciones del entorno. Se puede observar que el lóbulo central abarca un ángulo de ≈ 30 grados*

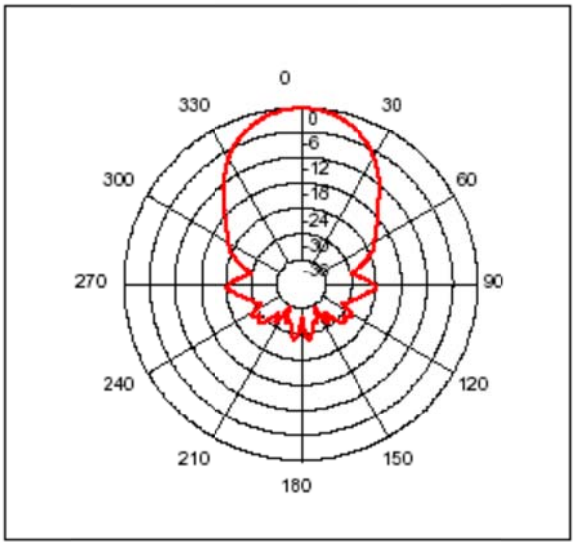


Figura 1.6 *Rango efectivo del sensor SRF04*

Existen básicamente tres posibles aproximaciones al problema de la medida de la distancia a un objeto:

- Los sensores basados en el tiempo de vuelo de un pulso de energía que viaja hacia un objeto en el que se refleja, para volver de nuevo al receptor.
- Los sensores basados en la medida del desplazamiento de fase, que necesitan una transmisión de señal continua, en lugar de emitir pulsos.
- Los sensores basados en un radar de frecuencia modulada. Esta técnica está bastante relacionada con la anterior.

El sensor utilizado en este proyecto está incluido dentro del primer grupo.

1.6.1 Sensores basados en el Tiempo de Vuelo

La mayor parte de los sensores usan esta técnica. El pulso de energía con el que se lleva a cabo la medida, puede provenir tanto de un generador de ultrasonidos, como de uno de energía óptica, o uno de radiofrecuencia.

Por lo tanto, debido a esto, los parámetros importantes para llevar a cabo la medición son la velocidad del sonido en el aire, y la velocidad de la luz. El tiempo medido es una representación de la distancia existente entre el dispositivo medidor, y el objeto que ha reflejado la onda.

La ventaja de estos sistemas, viene por su naturaleza directa de la medición, y del hecho de que un mismo dispositivo puede actuar tanto como generador, como receptor. La distancia al objetivo, puede obtenerse directamente de la salida del sensor, y no es necesario realizar ninguna suposición sobre si las superficies son planas, ni sobre la orientación existente entre éstas y el sensor.

Sus desventajas, vienen del hecho de que no se puede llevar a cabo un esquema de triangulación, debido a la nula distancia existente entre emisor y receptor. En cualquier caso, estos sistemas mantienen una buena precisión, siempre y cuando el obstáculo que se analiza presente un eco fiable.

Las fuentes de errores en la medida, vienen fundamentalmente de la imprecisión de la medida del tiempo, de las variaciones en la velocidad de onda transmitida, de las deficiencias en el circuito temporizador, o por interacción de la onda con la superficie en la que debe reflejarse. A continuación se incluye una breve descripción de cada uno de estos problemas:

- **Velocidad de propagación de la onda:** Para las aplicaciones de robótica móvil, las variaciones en la velocidad de propagación de las ondas electromagnéticas, puede ser totalmente ignorada. Este no es el caso de los sistemas acústicos, donde la velocidad del sonido, está muy influenciada por la temperatura y en menor medida por la humedad.
- **Incertidumbre de la detección:** Estas variaciones vienen motivadas por la distinta reflexión de las ondas en distintos tipos de objetos. Algunos las reflejan con una intensidad mayor que otros, lo que hace que los sistemas de detección respondan de forma más rápida ante los primeros, y por lo tanto hagan que esos objetos parezcan más próximos. Por este motivo, es muy importante la selección del valor umbral, a partir del cual se inicia la detección de onda.
- **Incertidumbre en la medida del tiempo:** Debido a la relativamente lenta velocidad del sonido en el aire, en comparación con la de la luz, los sistemas acústicos, tienen unos requisitos de velocidad muy inferiores a los de sus contrapartidas basadas en la luz. La velocidad a la que se transmiten las ondas electromagnéticas, obliga a imponer en los circuitos electrónicos unos requisitos de velocidad muy fuerte, requiriendo circuitería que responda en tiempos inferiores al nanosegundo. Para obtener precisiones de tan sólo 1 cm, se necesita una circuitería con temporizaciones de 3 nano-segundos. Conseguir esto resulta muy caro, y aparta a estos sistemas de medida de las aplicaciones normales.
- **Interacción con las superficies:** Cuando la onda se refleja en un objeto, sólo una pequeña fracción de la señal vuelve al receptor. El resto de la energía se refleja en otras direcciones, o es absorbida por la propia superficie. La energía reflejada en distintas direcciones, puede a su vez, volver a reflejarse en más objetos, y llegar finalmente al receptor siguiendo una línea que no es recta, o volver reflejada a un receptor que no corresponde con el lugar desde el que fue emitida.

1.7 El Sistema Ultrasónico

El sistema se compone fundamentalmente de un transductor electrostático, y de una pequeña placa electrónica en la que está la circuitería encargada de emitir los pulsos, recibirlos y procesarlos. Para determinar la distancia a un objeto, se mide de forma externa, el intervalo transcurrido entre la emisión y la recepción del pulso ultrasónico.

1.7.1 Descripción General del Sistema

La determinación de distancias por medio de un sistema de medición de ecos es un proceso muy simple. Un pulso corto de energía ultrasónica se genera de forma electrónica, se amplifica y se envía a un transductor. La señal viaja a través del medio (en general el aire), se refleja en un objeto y vuelve al transductor. Esta señal se recibe, se amplifica y se procesa por el sistema electrónico. El tiempo que ha tardado en viajar la señal se puede usar para determinar la distancia a la que está el objeto, al conocer la velocidad a la que se transmite el sonido en el aire.

El sistema de medida más simple se compone tan sólo de dos módulos, el módulo electrónico y el transductor. El transductor, de tipo electrostático o piezoeléctrico, se usa tanto para emitir el pulso, como para recibir el eco y procesar la información obtenida a éste.

La distancia del transductor al objetivo, puede determinarse por un circuito adicional que conozca la velocidad del sonido en el aire y el intervalo de tiempo transcurrido entre la emisión y la recepción de la señal.

Esta amplificación llevada a cabo en el receptor, es la parte más complicada del sistema, ya que debe ser de carácter exponencial (al igual que lo es la atenuación del sonido en el aire), y tiene que partir desde valores de amplificación pequeños hasta subir la intensidad de la señal recibida en varios ordenes de magnitud.

Para minimizar el peligro de que pequeños ruidos confundan al sistema y éste no crea recibir falsos ecos, el sistema dispone de un integrador, en el que se recibe la señal entrante. Sólo cuando a la salida del integrador se alcance un nivel determinado, se considera que ha recibido un eco auténtico.

Las señales de salida, indican los instantes en los que se envió el pulso, y el instante en el que se recibe el eco, por lo que resulta sencillo llevar a cabo funciones de control, así como medir el tiempo transcurrido entre ambos eventos.

1.7.2 El Transductor

La parte más importante del sistema es el transductor electrostático. Está compuesto por una membrana muy fina, una lámina recubierta de oro para formar el electrodo negativo de un diafragma de vacío. El electrodo positivo, es una lámina recubierta de aluminio, que también sirve como estructura resonante para el diafragma.

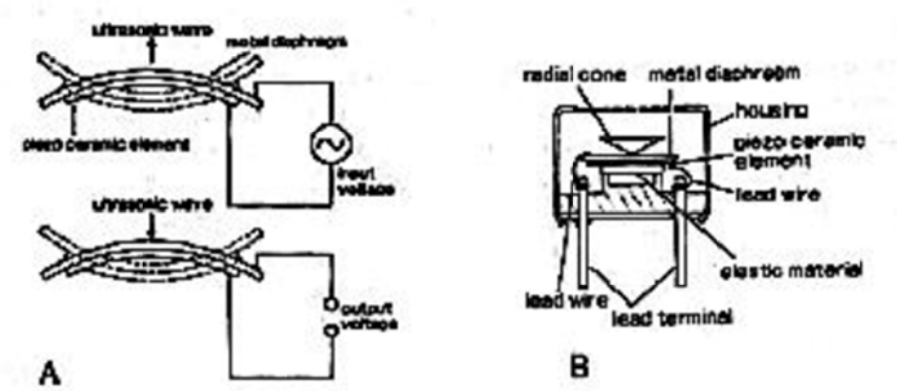


Figura 1.7 *Transductor del sensor de ultrasonidos*

La señal no se transmite de forma lineal en línea recta, a partir del transductor, sino que en lugar de esto se genera una señal que se extiende formando un cono.

1.7.3 El Módulo Electrónico

La electrónica del sistema de sonar está reunida en un único módulo, usando circuitos integrados específicos para realizar las funciones digitales y analógicas. Estos dos circuitos integrados, están unidos junto con los componentes discretos necesarios, en una pequeña placa de tan solo 20 mm de ancho y 43 mm de largo.

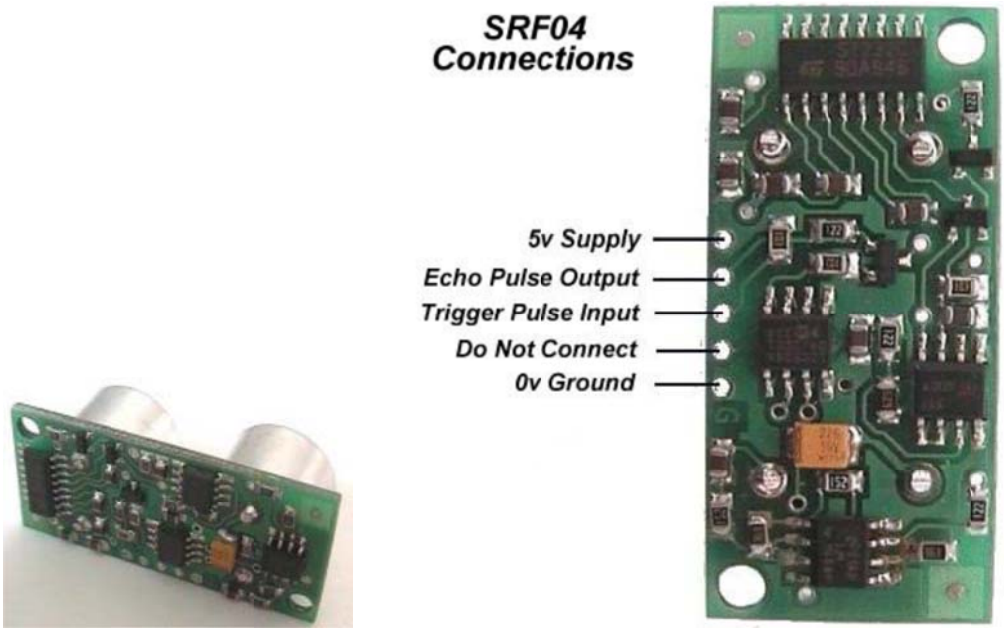


Figura 1.8 *Módulo del sensor de ultrasonidos SRF04*

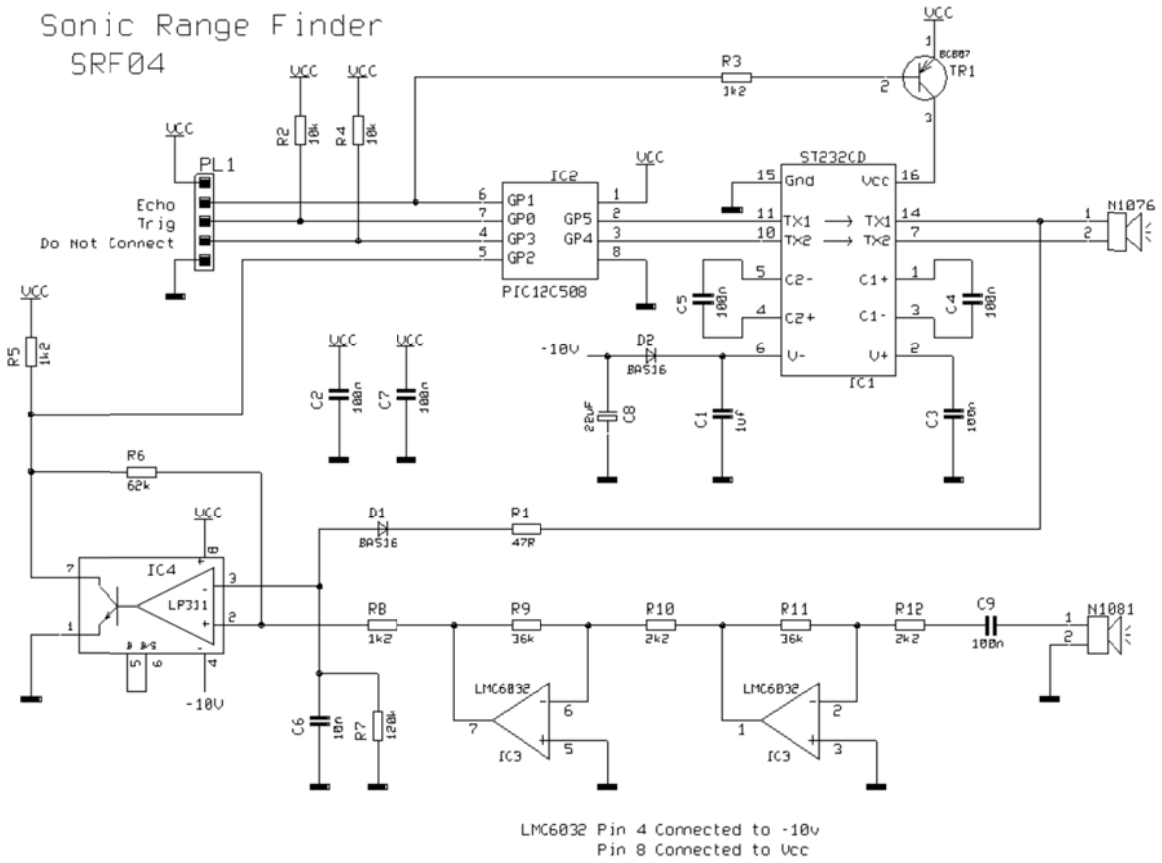


Figura 1.9 *Esquema del circuito que forma el módulo electrónico del sensor SRF04*

Cuando se activa el sistema, estos circuitos generan una serie de pulsos a frecuencias discretas distintas. La señal se amplifica mediante la circuitería ya incorporada en la placa. La componente continua se mantiene en el transductor, por medio de un condensador de almacenamiento, mientras éste actúa como micrófono para recibir el eco. El transductor se bloquea durante unos pocos microsegundos, para evitar la recepción del eco que se genera al salir la señal del propio transductor, y a partir de ese instante pasa a actuar como un micrófono.

Para compensar la pérdida de energía de la señal al recorrer mayores distancias, la ganancia del amplificador que trabaja en la recepción de la señal, se va incrementando en función del tiempo, además de disminuir en ancho de banda para disminuir los efectos del ruido.

Cuando una señal es recibida por el sistema, y si esta señal supera un nivel umbral mínimo, se activa una fuente de corriente, que va cargando un condensador hasta que alcance éste los 1,2V. En este instante, se considera que la señal recibida es el eco y se genera una señal lógica.

Para medir el intervalo de tiempo transcurrido entre las dos señales hay que recurrir a un circuito externo. Al multiplicar este valor por la velocidad del sonido, y teniendo en cuenta que la se ha recorrido dos veces. Dividiendo entre dos, se obtiene la distancia a la que se encuentra el objeto del transductor.

1.7.4 Medida del Eco

Mientras que el hecho de medir el eco recibido es relativamente simple, algunos de los procesos involucrados en él no lo son. La propagación de la energía acústica a través de un medio fluido es muy compleja, pero afortunadamente, a las frecuencias con las que se trabaja está bastante bien estudiado, y la mayor parte de los factores de atenuación, reflexión, refracción, etc. pueden ser simplificados de una forma bastante satisfactoria.

Los factores acústicos que más afectan al rendimiento del sistema de medida por sonar están relacionados con el rendimiento del transductor, su frecuencia de trabajo y la distancia máxima a la que se desee llegar con las medidas.

Obviando los aspectos electrónicos, el aspecto que más interesa es la relación existente entre el alcance y la frecuencia de trabajo. Ésta es debida a la atenuación que tiene cada frecuencia al viajar por el aire. El mecanismo de esta pérdida es bastante complejo, dependiendo de factores tales como la temperatura, la humedad, etc. El efecto de atenuación debido al a humedad es máximo, al trabajar con valores intermedios de la misma.

1.7.5 Resolución

En general, con un sistema de medida de ecos ultrasónicos, se puede llegar a alcanzar una resolución del orden de la longitud de onda de la señal que se transmite.

Esto depende del tipo de sistema de detección empleado. En estos sensores se usa una sencilla técnica de integración. Cuando la señal que se recibe, supera un umbral determinado, se activa una fuente de corriente que empieza la carga de un condensador. Existe un drenaje continuo de la carga de este condensador, para garantizar que pequeños picos de ruido consecutivos no vayan cargando el condensador, y lo lleven al valor umbral en el que se identifica la señal como positiva.

Usando una velocidad de carga de este condensador, diez veces superior a la velocidad de descarga, se consigue un esquema que puede alcanzar una resolución de aproximadamente unos 6mm, siempre y cuando el objetivo esté fijo respecto al sensor, y se cumplan algunas otras restricciones.

Si se necesita alcanzar realmente esta precisión máxima, es necesario conocer de forma precisa la velocidad del sonido en el aire. La velocidad del sonido en el aire, puede ser determinada de acuerdo con la expresión:

$$v = 331,4 \cdot \sqrt{T/273} \qquad (1.4)$$

Donde T es la temperatura en grados Kelvin y se ve que existe una fuerte dependencia de la temperatura.

Si se trabaja dentro de un margen de temperaturas de entre -30ºC y +30ºC, entonces esta expresión puede aproximarse por:

$$v = 331,4 + 0,607 \cdot t \qquad (1.5)$$

Donde t es la temperatura en grados centígrados.

La dependencia de la temperatura es muy fuerte, y por lo tanto si se desea obtener una precisión alta, es necesario llevar a cabo la compensación. En un entorno en el que la temperatura pueda oscilar en un rango de 60ºC, la variación de la velocidad del sonido llega a un 5%. La corrección de este valor puede llevarse a cabo bien por el sistema de medición, o bien por el procesador que vaya a analizar los datos obtenidos.

1.8 Funcionamiento del Sensor

El funcionamiento típico de un sensor de ultrasonidos viene dado por un ciclo de operación que sigue los siguientes pasos:

1. El circuito de control dispara el transductor, quedando a la espera de una señal que confirme el comienzo de la transmisión.
2. El circuito de recepción es blanqueado durante un tiempo, para evitar que ondas residuales de la transmisión puedan ser interpretadas como falsos ecos.
3. Las señales recibidas son amplificadas mediante un amplificador de ganancia variable, el cual compense la atenuación del medio en aquellas señales que han recorrido una mayor distancia.
4. Las señales recibidas que superen un determinado nivel son reconocidas como ecos, calculándose la distancia a la que se encuentra el objeto que lo ha provocado.

De los puntos anteriores se deduce que la distancia mínima a la que un sensor responderá, vendrá dada por el tiempo de blanqueo. Para el caso de nuestro sensor utilizado, según los datos aportados por el fabricante, el tiempo es aproximadamente 300µs. Sustituyendo este valor en la ecuación [1.3] se obtiene una distancia mínima de ≈ 5cm, habiendo considerado una temperatura ambiente de 25 grados. No obstante, esto es un valor teórico, por lo tanto esta distancia mínima puede variar debido a las no-idealidades de los dispositivos electrónicos que se encargan de procesar la señal.

Por otro lado, la distancia máxima vendrá dada por la atenuación de la onda ultrasónica en el medio en el que se propague (aire) y por la ganancia del amplificador que recoge los ecos.

La señal emitida por el sensor consta de 8 pulsos a 40KHz y la apertura del cono de emisión de la onda ultrasónica se calcula a partir de la siguiente expresión:

$$\vartheta = \arcsin\left(\frac{0{,}61 \cdot \lambda}{r}\right) \quad \textit{(1.6)}$$

Donde λ es la longitud de onda y r es el radio del anillo exterior del sensor.

Los sensores de ultrasonidos tienen un transductor acústico que vibra a frecuencias ultrasónicas. Los pulsos son emitidos en haz cónico y apuntan a un objeto. Los pulsos se reflejan en el objeto y vuelven al sensor en forma de ecos. El instrumento de medida mide el tiempo de retraso entre cada emisión y el pulso de eco para determinar exactamente la distancia del sensor al objeto. Los sensores de ultrasonidos detectan todos los objetos, sean del material que sean, independientemente del color que tengan. Detectan objetos claros, transparentes y brillantes tan fácilmente como los oscuros y opacos. Esta habilidad permite a los sensores de ultrasonidos detectar todo el rango de materiales, desde una botella de cristal transparente hasta neumáticos de goma negra. Si un material se cubre, el sensor puede detectar de manera exacta y repetidamente el material cubriente a pesar de los brillante o claro que sea (no como ocurre con los sensores infrarrojos). Los sensores de ultrasonidos funcionan bien en ambientes toscos (humos, polvo, ruido).

1.9 Incidencias del Medio Ambiente

1.9.1 Temperatura

La velocidad del sonido en el aire depende de la temperatura. Un sensor interno de temperatura puede adaptar la frecuencia del reloj del contador y la frecuencia de la portadora para ayudar a compensar las variaciones de la temperatura del aire. Sin embargo, las fluctuaciones grandes de temperatura dentro del camino que recorre la onda ultrasónica pueden causar dispersión y refracción de la señal de ultrasonidos, afectando negativamente a la exactitud y estabilidad de la medida. Si se quiere detectar un objeto caliente, habrá que experimentar posicionando el sensor y el objeto en un plano vertical y apuntar a la parte del objeto más fría. De esta manera, se pueden evitar las corrientes de aire caliente y lograr la operación satisfactoriamente.

1.9.2 Presión del Aire

Los cambios normales en la presión atmosférica del aire no tienen efectos sustanciales en la exactitud de la medida. Pero se aconseja no usar sensores de ultrasonidos con presiones de aire bajas o demasiado elevadas.

1.9.3 Humedad

El efecto de la humedad en la medida es virtualmente insignificante, sólo cambia un 0,07% para un cambio en la humedad relativa de 20%. Sin embargo, la absorción del sonido aumenta con el aumento de la humedad. Así, la máxima distancia de medida es reducida muy poco.

1.9.4 Turbulencias en el Aire

Las corrientes de aire, turbulencias, y capas de distinta densidad causan refracción de la onda de sonido. Un eco puede ser producido y la señal debilitada o desviada, y por lo tanto el eco no es recibido. El máximo rango sensible, la precisión de medida y la estabilidad de medida pueden deteriorarse bajo estas circunstancias.

1.10 Errores de Medida con Ultrasonidos

La medida de distancias con ultrasonidos está sometida a diferentes fuentes de error, los más comunes son los detallados a continuación.

1. Errores de origen natural

    - La velocidad de propagación del sonido en el aire depende de la temperatura. (Mostrada en la ecuación [1.2])

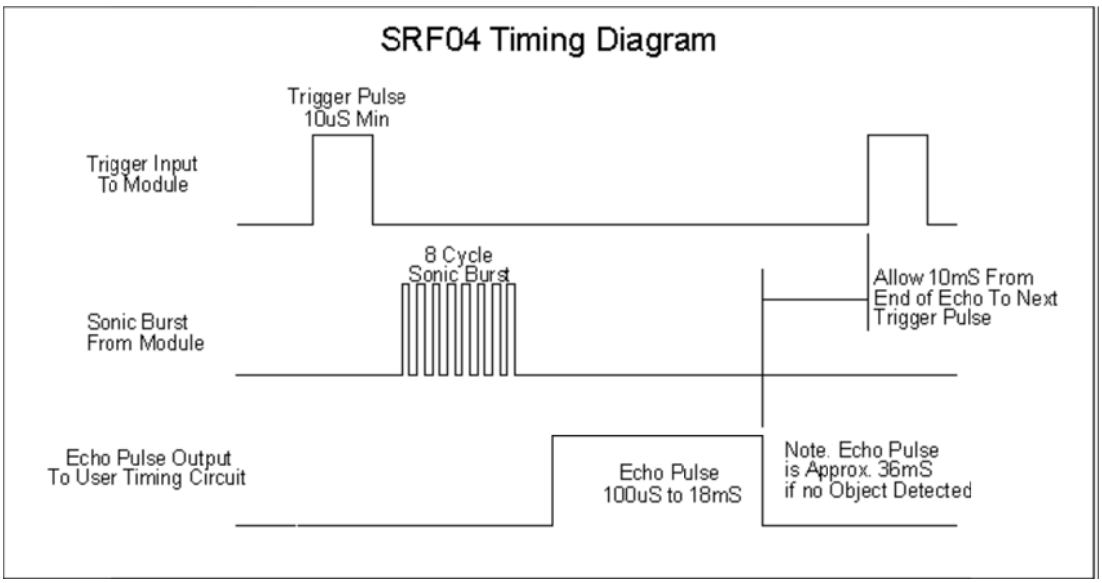


Figura 1.10 *Cronograma del proceso de emisión de una onda ultrasónica mediante el sensor de ultrasonidos SRF04*

    - Según sea la rugosidad de los objetos, éstos serán reflectantes o refractantes. Aunque este efecto se corrige con la emisión de un pulso de varias frecuencias, existen objetos que pueden atrapar la onda, devolviendo una distancia mayor de la que en realidad se encuentra.
    - El patrón de emisión de la onda no es completamente cónico (Ver figura 1.5 y 1.6), si no que existen unos lóbulos laterales de emisión que pueden provocar un eco.

2. Errores de origen electrónico

    - El amplificador de ganancia variable que contrarresta el efecto de la atenuación en el aire es aproximado por 16 niveles de amplificación. Los puntos de discontinuidad en esta gráfica provocan un error en la corrección.
    - El nivel que determina si un eco es reconocido como tal se realiza mediante la carga de un condensador. Los ecos de objetos cercanos cargan este condensador en tres periodos, sin embargo los ecos de objetos más lejanos necesitan más periodos, por lo que aparentarán encontrarse a una distancia mayor.

3. Errores debidos a la posición relativa Sonar-Objeto

- Según sea la posición relativa de los sónares frente a un objeto, se pueden producir diferentes efectos que pueden hacer que un objeto que esté más cercano aparezca como más lejano o viceversa, que sea inapreciable. La secuencia de gráficos de la siguiente figura muestra alguno de los errores más comunes.

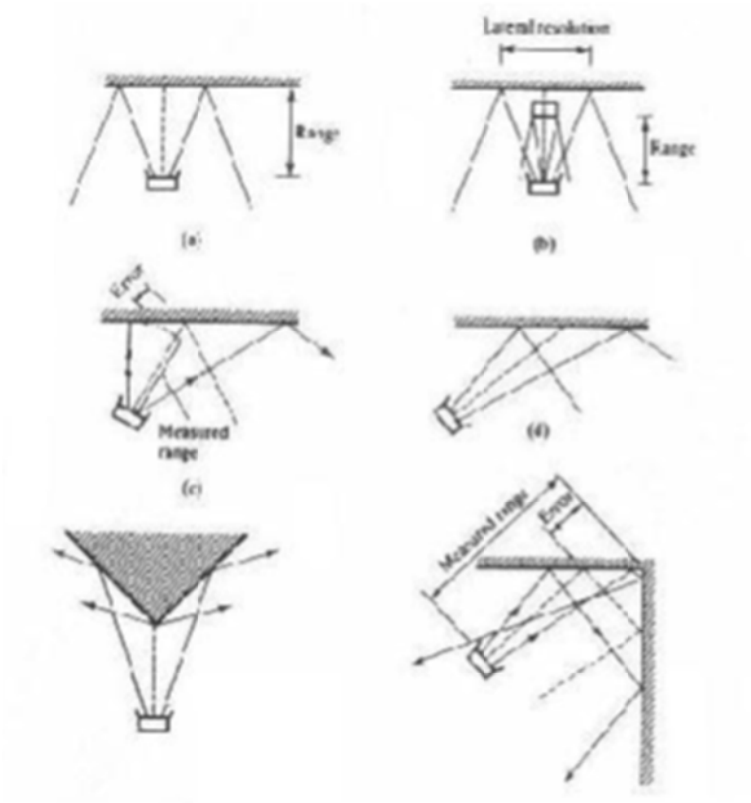


Figura 1.11 *Posibles errores debido a la disposición relativa entre el sonar y el objeto. La figura (**a**) muestra una medida de distancia correcta. En la figura (**b**) se muestra el problema de no poder determinar el ancho del objeto que refleja la onda, debido a la apertura del cono. En la figura (**c**) se obtiene una distancia más corta de la que en realidad existe al objeto, debido a que el eco producido se debe al extremo del cono y no a la parte central. En la figura (**d**) se muestra un objeto inapreciable, ya que los ecos no regresan al punto de emisión. La figura (**e**) muestra el mismo problema para el caso de una esquina. En la figura (**f**) la medida obtenida es mayor de la real debido a los falsos ecos producidos por sucesivos rebotes.*

1.11 Aplicaciones

1.11.1 Introducción

Los sensores de ultrasonidos proporcionan al mercado un método de detección eficaz a bajo coste con unas propiedades únicas que no poseen otras tecnologías de detección. Usando una gran variedad de transductores ultrasónicos y varios rangos distintos de frecuencia, un sensor de ultrasonidos puede ser diseñado para resolver muchos problemas de aplicación que no se pueden hacer por su coste elevado o simplemente porque no pueden resolverse mediante otros sensores.

Gran Rango de Detección: En la detección industrial mediante sensores, muchísimas aplicaciones requieren detección a larga distancia. Los sensores de ultrasonidos detectan a distancias superiores a los cuarenta pies, mientras que los sensores inductivos no lo pueden hacer.

Área de Detección Ancha: Mientras que algunos sensores fotoeléctricos pueden detectar a largas distancias, carecen de la habilidad de detectar sobre áreas grandes sin usar un número elevado de sensores. La ventaja de los sensores de ultrasonidos es que pueden cubrir tanto áreas anchas como estrechas.

Permiten Detectar todo tipo de Materiales: Solamente los sensores de ultrasonidos están capacitados para detectar todo tipo de materiales, sea cual sea su composición. El material detectado puede ser claro, sólido, líquido, poroso, blando, madera y de cualquier color porque todos son detectados.

Miden Distancias sin Necesidad de Contacto: Se mide el tiempo que tarda el sonido desde que deja el transductor hasta que vuele a él, por lo que la medida de la distancia es sencilla y exacta con un margen de error de 0,05%.

1.11.2 Ventajas

Las ventajas de estos sensores con respecto a otros son:

- Miden y detectan distancias a objetos en movimiento.
- Es independiente el tipo de material, superficie y color del objeto a detectar.
- Detectan objetos pequeños a distancias grandes.

- Son resistentes frente a perturbaciones externas tales como vibraciones, radiaciones infrarrojas, ruido ambiente y radiación EMI (ruido que se origina de una fuente y viaja a través del aire).
- No son afectados por el polvo, suciedad o ambientes húmedos.
- No es necesario que haya contacto entre el objeto a detectar y el sensor.

1.11.3 Diferencia entre Detección de Proximidad y Medida del Rango

El sensor de ultrasonidos se puede usar para:

La Detección de Proximidad: Un objeto pasando por algún sitio dentro del rango de alcance presente del sensor puede ser detectado y generar una señal de salida. El punto de detección es independiente del tamaño del objeto, de su material o del grado de reflexión. Las aplicaciones de detección de proximidad incluyen la detección de la presencia o ausencia de personas y objetos de interés. El sensor detectará el objeto que esté dentro del cono de emisión y reflexión de la señal de ultrasonidos. La detección de objetos situados a grandes distancias requiere que el objeto sea grande o esté orientad de modo que la señal que refleja llegue al ultrasonido.

La Medida de la Distancia: La distancia precisa a un objeto en movimiento del sensor se mide vía el intervalo de tiempo entre la señal ultrasónica transmitida y la recepción de la reflejada. El ejemplo muestra la detección de un objeto que está a 6 pulgadas (15,24 cm) y al rato, al moverse, está a 10 pulgadas (25,4 cm). La distancia cambiante se calcula continuamente.

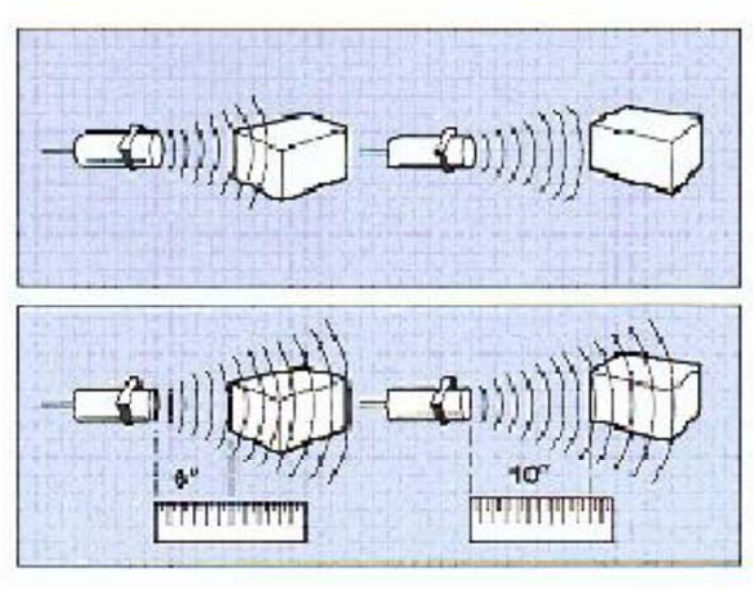


Figura 1.12 *Ejemplo de la medición con un objeto en movimiento*

1.11.4 Campos de Aplicación

Máquinas Constructoras: Los sensores de ultrasonidos son usados para el control del movimiento, el control del nivel o para dimensionar o detectar proximidad en máquinas que se usan en construcción. Estas son comunes en aplicaciones dentro de la industria de transformación, impresión, de los metales, la textil y otras industrias de manufacturación.

Automatización: Los sensores de ultrasonidos reducen los costes de automatización proporcionando métodos de control de tamaños y detección de posición o proximidad de objetos en los procesos de producción. La información proporcionada por los sensores es usada para:

- Aceptar o rechazar objetos basándose en el tamaño, posición o nivel de llevando.
- Para tomar decisiones sobre el camino que deben seguir los paquetes en el tamaño o posición.
- Controlar el flujo de líquidos, sólidos o materiales granulados.
- Indicar cuando un objeto está cercano o en la posición debida.
- Determinar tolerancias.
- Proporcionar una señal de alarma cuando los objetos no están en la posición debida, están a punto de llenarse o vaciarse.
- Para indicar la terminación de un proceso.

Control de Procesos: Las aplicaciones comunes incluyen medida del nivel de materiales en un tanque o lata, o controlar la cantidad de material dispersado desde un contenedor. Se puede dar cuenta de la medida del nivel de un tanque a una computadora mediante una red de datos. Las alarmas pueden dispararse debido a un nivel bajo, un determinado nivel establecido, un nivel elevado u otras condiciones.

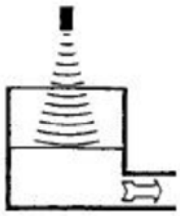
Información y Diversión: Se puede detectar a gente que se aproxima a una cabina, o se puede controlar su distancia para establecer una determinada respuesta. También se puede usar la detección de personas y objetos en determinados juegos.

Derroche de Agua: Se pueden medir los niveles de depósitos de agua para controlarlos. El flujo de agua de un canal abierto se mide para dar informes y establecer un control.

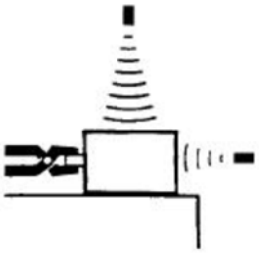
1.11.5 Aplicaciones Típicas

- Para la medida del diámetro de un arrollamiento, para control de enrollado o desenrollado, o para medida de la tensión de una tela. Esto es usado en la fabricación de papel o tela, la fabricación de gomas o neumáticos, el procesado de acero, etc.

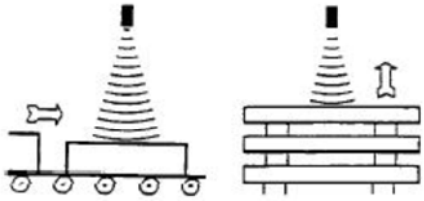
- Para el control de nivel de tanques (tanto de líquidos como de granulados), o para el control del nivel de llenado de botellas o latas. Usado en la industria alimenticia, química o de plásticos.



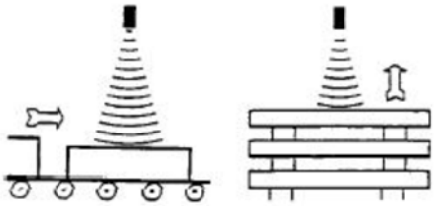
- Para la medida de distancia, para el posicionamiento de piezas de trabajo en robots y para la medida de tamaño (dimensionamiento). Se usa en la industria de la automatización, el trabajo con metales y en equipos de ensamblaje.



- Para la medida de tamaño o anchura o para el empaquetamiento. Se usa para el manejo de materiales o para trabajar con metales.



- Para la detección de presencia o ausencia, o para la detección de partes claras o de cristal o vidrio. Se usa en la industria alimenticia, manejo de materiales o para equipos de ensamblaje.



- Para el control de movimiento, detección de personas, seguridad, etc.

Capítulo 2. Estudio del Microcontrolador

2.1 Microcontroladores PIC

Un microcontrolador es un circuito integrado programable que contiene todos los componentes necesarios para controlar el funcionamiento de una tarea determinada, como por ejemplo un sistema de alarmas. Para esto, el microcontrolador utiliza muy pocos componentes asociados. Un sistema con microcontrolador debe disponer de una memoria donde se almacena el programa que gobierna el funcionamiento del mismo, que una vez programado y configurado, sólo sirve para realizar la tarea asignado. La utilización de un microcontrolador en un circuito reduce considerablemente el tamaño tamaño y número de componentes y, en consecuencia, disminuye el número de averías, el volumen y el peso de los equipos, entre otras ventajas.

Son muchos los fabricantes que fabrican este tipo de dispositivo, para la realización del proyecto se ha elegido los microcontroladores PIC (*Peripheral Interface Controller*). Son una familia de microcontroladores que ha tenido gran aceptación y desarrollo en los últimos años gracias a sus buenas características, bajo precio, reducido consumo, pequeño tamaño, gran calidad, fiabilidad, abundancia de información y sobre todo por su comodidad y sencillez de utilización. Son fabricados por *Microchip Technology Inc*.

2.2 PIC16F84A

Para la realización del proyecto se ha optado por el PIC16F84A, dicho microcontrolador de 8 bits es suficientemente potente para realizar la aplicación necesaria de medir distancias. Está encapsulado en un DIL de 18 pines (figura 2.19). Debido a sus múltiples aplicaciones, precio, fácil obtención y facilidad de uso, es uno de los microcontroladores más utilizados a la hora de realización de proyectos.

El PIC16F84A se alimenta con 5 voltios aplicados entre los pines V_{DD} y V_{SS}, y puede llegar a trabajar hasta una frecuencia de reloj de 20Mhz, en nuestro proyecto hemos utilizado una frecuencia de reloj de 4MHz por la facilidad de encontrar un cristal de cuarzo de 4MHz y sobre todo por la facilidad de cálculos a la hora de realizar los retardos en la programación, que se explicará más adelante. Por estos motivos también se ha elegido es microcontrolador.

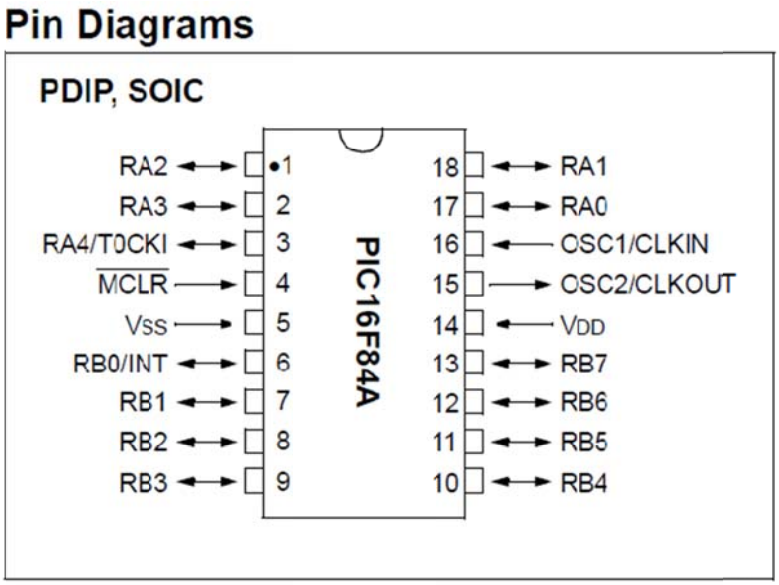


Figura 2.1 *Encapsulado DIL-18*

2.3 Arquitectura del PIC16F84A

Los microcontroladores PIC utilizan una arquitectura HARVARD que dispone de dos memorias independientes a las que se conecta mediante dos grupos de buses separados:

- Memorias de datos.
- Memoria de programa.

Ambos buses son totalmente independientes y pueden ser de distintos anchos, esto permite que la CPU pueda acceder de forma independiente y simultánea a la memoria de datos y a la de instrucciones, consiguiendo que las instrucciones se ejecuten en menos ciclos de reloj.

Esta dualidad de la memoria de datos por un lado y por otro la memoria de programa permite la adecuación del tamaño de las palabras y los buses a los requerimientos específicos de las instrucciones y los datos.

Se puede concluir que las principales ventajas de la arquitectura Harvard son:

- El tamaño de las instrucciones no está relacionado con el de los datos y, por lo tanto, puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa. Así se logra una mayor velocidad y una menor longitud de programa.

- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

A continuación, en la figura 2.2 se muestra el diagrama de bloques de la arquitectura interna del PIC16F84A.

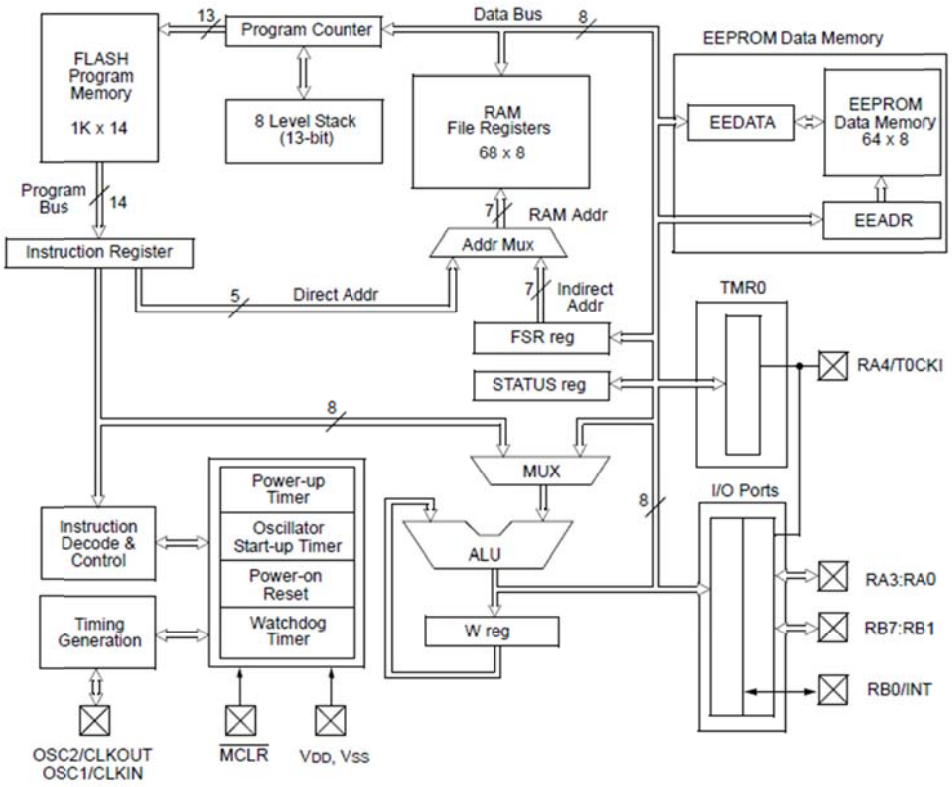


Figura 2.2 *Estructura interna del PIC16F84A*

2.3.1 Breve Explicación de los Bloques que forman la CPU del PIC16F84A

2.3.1.1 Registro de Trabajo W

El registro de trabajo W (Work Register) es el registro principal y participa en la mayoría de las instrucciones. Se localiza dentro de la CPU del PIC16F84A como se aprecia en la figura 2.2.

2.3.1.2 ALU (*Arithmetic Logic Unity*)

En la misma figura se muestra como el microcontrolador posee una ALU (Arithmetic Logic Unity) de 8 bits. Esta se encarga de realizar las operaciones lógicas o aritméticas que requiere la ejecución del programa con dos operandos, uno que proviene del registro W y el otro que se encuentra en cualquier otro registro o en el propio código de instrucción.

En los microcontroladores PIC la salida de la ALU (Arithmetic Logic Unity) va al registro de trabajo denominado W (Work Register) y también al resto de bloques que están comunicados con la ALU, así el resultado puede guardarse en cualquiera de los dos destinos. En las instrucciones de doble operando, uno de los dos datos siempre debe estar en el registro W. La gran ventaja de esta arquitectura es que permite un gran ahorro de instrucciones ya que el resultado de cualquier instrucción que opere con la memoria puede dejarse en la misma posición de memoria o en el registro W.

2.3.1.3 Memoria de Programa

El mircrocontrolador está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. El programa a ejecutar siempre es el mismo, por lo tanto, debe estar grabado de forma permanente. Esta característica de “no volatilidad” garantiza que la memoria mantenga su contenido aún sin alimentación, de forma que el programa no necesite volver a ser cargado en el sistema cada vez que se utilice.

La información contenida en estas memorias debe ser grabada previamente mediante un equipo físico denominado programador o grabador. Este equipo se debe conectar a un ordenador que mediante un software controla la grabación de la memora de programa del microcontrolador.

El PIC16F84A es un microcontrolador con un tipo de memora de programa no volátil denominada ROM Flash, que permite una cómoda grabación, lo que representa una gran facilidad en el desarrollo de diseños. La memoria del programa del PIC16F84A tiene una capacidad de 1k (1024 posiciones) y está organizada en palabras de 14 bits. La memoria de programa comienza en la posición 000h (posición inicial de reset) y llega hasta la 3FFh.

2.3.1.4 El Contador de Programa (*Program Counter*)

Un programa está compuesto por instrucciones que generalmente se ejecutan de forma secuencial. En el PIC16F84A cada una de esas instrucciones ocupa una posición de memora de programa.

El contador de programa o PC (*Program Counter*) es un registro interno que se utiliza para direccionar las instrucciones del programa de control que están almacenadas en la memoria de programa (ver figura 2.2). Este registro contiene la dirección de la próxima instrucción a ejecutar y se incrementa automáticamente de manera que la secuencia natural de ejecución del programa es lineal, una instrucción después de otra.

El microcontrolador PIC16F84A dispone de un contador de programa que le permite direccionar los 1k x 14 bits de memoria de programa implementada, desde la posición 000h hasta la 3FFh.

2.3.1.5 Memoria de Datos

En esta memoria se almacenan los datos que se manejan en un programa. Estos datos varían continuamente, por lo que esta memoria debe ser de lectura y escritura. Se utiliza memoria denominada RAM. Esta memoria es de tipo volátil, los datos se borran en caso de que desaparezca la alimentación.

Las tablas 2.1 y 2.2 muestran los registros ubicados en de la memora de datos RAM del PIC16F84A donde se aprecia que está dividida en dos partes:

- Registros de Funciones Especiales SFR (Special Function Registers). Son los primeros registros, cada uno de ellos cumple un propósito especian en el control del microcontrolador.
- Registros de Propósito General GPR (General Purpose Registers). Son registros de uso general que se pueden usar para guardar los datos temporales del programa que se esté ejecutando. Para el PIC16F84A tiene 68 posiciones.

La memoria de datos cuenta con dos bancos de memoria, Banco 0 y Banco 1:

- Los registros del SFR están agrupados entre las direcciones 00h a 0Bh para el Banco 0 y entre las direcciones 80h hasta 8Bh para el Banco 1. Algunos de los registros del SFR se encuentran duplicados en la misma dirección en los dos bancos, con el objeto de simplificar su acceso.

- El banco de registros de propósito general está formado por 68 posiciones de memoria, ya que sólo son operativas las del Banco 0 (direcciones desde la 0Ch hasta la 4Fh), porque las del Banco 1 se mapean sobre el Banco 0. Es decir, cuando se apunta a un registro de propósito general del Banco 1 (direcciones de 8Ch hasta 0CFh), realmente se accede al mismo registro del Banco 0.

Para seleccionar el banco a acceder hay que configurar el bit 5 (RP0) del registro STATUS. Con RP0 = 0 se accede al Banco 0 y con RP0 = 1 se accede al Banco 1. El Banco 0 es seleccionado automáticamente después de un reset.

Las zonas de memoria 50h-7Fh y D0h-FFh no son empleadas y devuelven 0 en caso de lectura.

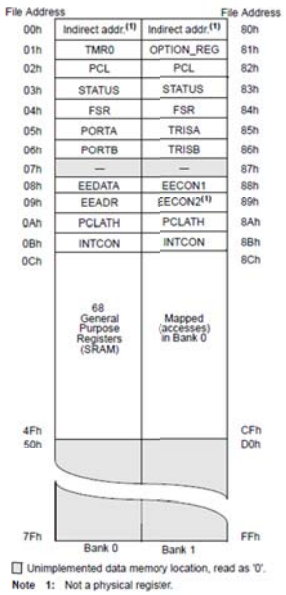


Tabla 2.1 *Mapa de Registros del PIC16F84A*

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.

3: Other (non power-up) RESETS include: external RESET through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

Tabla 2.2 *Registros de Funciones Especiales (SFR)*

2.3.1.6 Puertos

El PIC16F84A dispone de dos puertos paralelos A y B. Las líneas de estos puertos se pueden programar individualmente como entradas o como salidas, y se utilizan casi de la misma forma. Debido al escaso encapsulado, con sólo 18 pines, determinadas líneas de estos puertos se comparten en otros recursos internos. A continuación se muestra una breve explicación de la constitución interna de éstos.

PUERTO A

El puerto A está constituido por 5 líneas RA4:RA0 cuyo sentido de trabajo es controlado mediante el registro TRISA, en el que un bit a “0” configura la línea correspondiente como salida, y un bit a “1” como entrada. Después de un reset, todos los bits del registro TRISA quedan a uno, por lo que todas las líneas del Puerto A quedan configuradas como entradas.

Estas líneas son capaces de entregar niveles TTL cuando la tensión de alimentación en V_{DD} es de 5V. Cada línea de salida puede suministrar una corriente máxima de 20 mA cuando está a nivel alto o absorber una corriente máxima de 25 mA cuando está a nivel bajo. Como hay una limitación de disipación máxima de potencia del chip, está limitada la suma de corriente por las cinco líneas del Puerto A, que no puede exceder de 50 mA cuando están a nivel alto y 80 mA cuando están a nivel bajo.

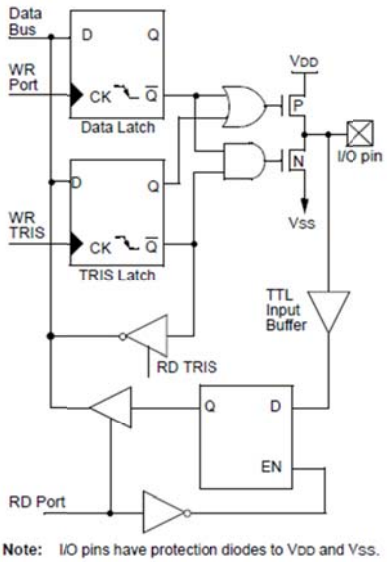


Figura 2.3 *Constitución interna de los pines RA3:RA0*

La línea RA4 adopta una estructura diferente:

- Cuando se configura como salida tiene una configuración de tipo drenador abierto, por lo tanto, necesita una resistencia de Pull-Up externa.
- Cuando se configura como entrada está provista de un Trigger Schmitt que proporciona una buena inmunidad al ruido, por ello esta línea se debe utilizar preferentemente como entrada frente a cualquier otra. Es común con la entrada externa del Timer 0.

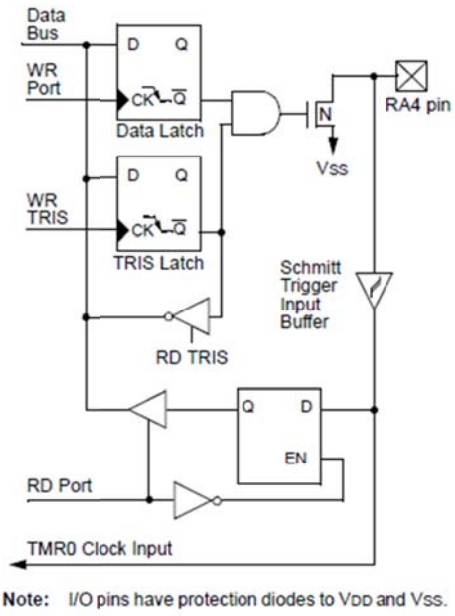


Figura 2.4 *Constitución interna del pin RA4*

PUERTO B

El puerto B es un puerto bidireccional de 8 bits completo, en el que sólo la línea RB0/INT tiene dos funciones multiplexadas, la propia entrada/salida del puerto y la petición de interrupción externa. Las líneas RB0 a RB3 adoptan una estructura distinta a las de las líneas RB4 a RB7 según se aprecian en las figuras 2.5 y 2.6. La razón de ser esta diferencia radica en el hecho de que es posible programar la generación de una interrupción durante un cambio de estado de una cualquiera de las líneas RB7:RB4. Todas las líneas del Puerto B disponen de una resistencia de Pull-Up de alto valor, conectada a la alimentación.

Las líneas RB7:RB4, cuando actúan como entradas, pueden ser programadas para generar una interrupción si alguna de ellas cambia su estado lógico. La figura muestra el diagrama interno de una de estas líneas. El circuito permite detectar la variación de una de estas señales cuando está en modo entrada, para ello, compara la última señal memorizada durante la última lectura del Puerto B. El cambio de una de las señales de entrada produce una interrupción que se refleja en el flag RBIF del registro INTCON.

Estas líneas son capaces de entregar niveles TTL cuando la tensión de alimentación aplicada en V_{DD} es de 5V. Cada línea de saluda puede suministrar una corriente máxima de 20 mA cuando está a nivel alto o absorber una corriente máxima de 25 mA cuando está a nivel bajo. Como hay una limitación de disipación máxima de potencia del chip, está limitada la suma de corriente por las ocho líneas del Puerto B, que no puede exceder de 100 mA cuando están a nivel alto y 150 mA cuando están a nivel bajo.

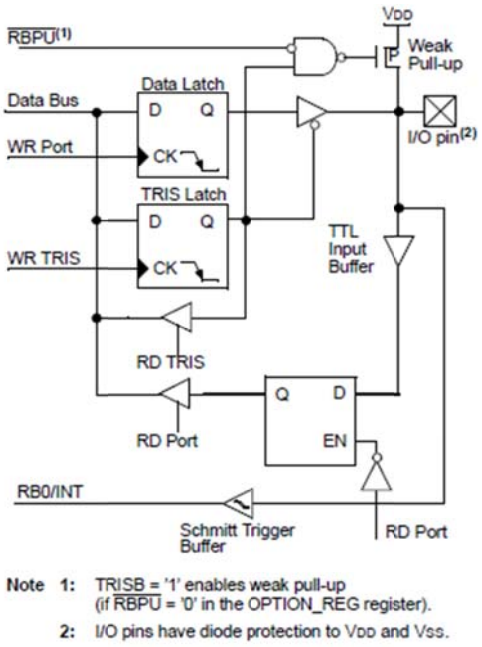


Figura 2.5 *Constitución interna de los pines RB3:RB0*

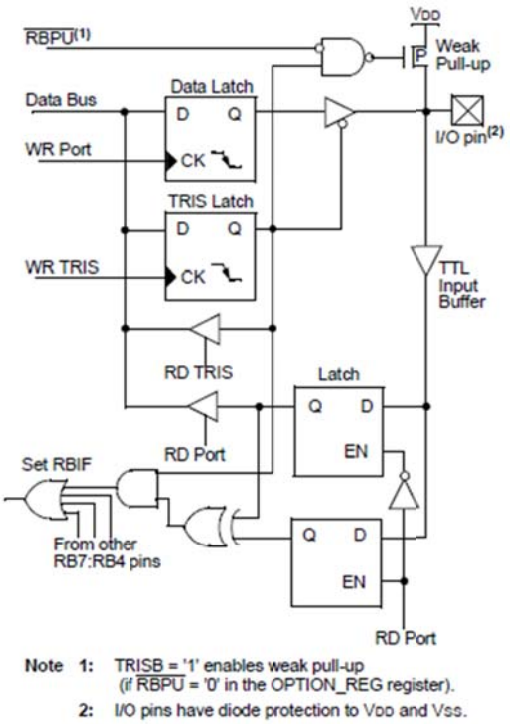


Figura 2.6 *Constitución interna de los pines RB7:RB4*

2.3.1.7 Memoria EEPROM

En ocasiones es necesario guardar la información que se genera durante el proceso de una forma permanente, es decir, esos datos han de permanecer incluso cuando el sistema se desconecta de la alimentación. El PIC16F84A dispone de una zona con 64 bytes de memoria EEPROM para almacenar datos que no se pierden al desconectar la alimentación. Esto es muy útil ya que permite guardar datos permanentemente. En la figura se muestra la estructura de esta memoria y los registros asociados.

Como en cualquier otra memoria EEPROM se pueden realizar dos tipos de operaciones:

- Operación de lectura.
- Operación de escritura o grabación.

Un ciclo de grabación en una posición EEPROM de datos dura unos 10 ms, un tiempo muy elevado para la velocidad del procesador, que se controla mediante un temporizador interno. Al escribir en una posición de memoria ya ocupada, automáticamente ya se borra el contenido que había y se introduce el nuevo dato, por lo que no hay comando de borrado.

El PIC16F84A soporta un millón de ciclos de escritura/borrado de su memoria EEPROM de datos y es capaz de guardar la información inalterada durante más de 40 años.

Esta memoria no forma parte del espacio direccionable y sólo es accesible para lectura y escritura a través de los registros mostrados en la tabla 2.3.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
08h	EEDATA	EEPROM Data Register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM Address Register								xxxx xxxx	uuuu uuuu
88h	EECON1	⊢	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM Control Register 2								---- ----	---- ----

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends upon condition.
Shaded cells are not used by data EEPROM.

Tabla 2.3 *Registros asociados con la memoria EEPROM*

2.3.1.8 El Timer 0 (TMR0)

Un timer se implementa por medio de un contador que determina un tiempo preciso entre el momento en que el valor es cargado y el instante en que se produce su desbordamiento.

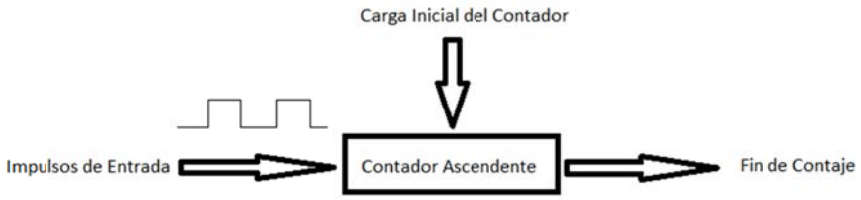


Figura 2.7 *Esquema simplificado de un timer*

El PIC16F84A dispone de un timer principal denominado *Timer 0* o *TMR0* que es un contador ascendente de 8 bits. El TMR0 se inicializa con un valor, que se incrementa con cada impulso de entrada hasta su valor máximo b'11111111'; con el siguiente impulso de entrada el contador se desborda pasando a valer b'00000000', circunstancia que se advierte mediante la activación del flag de fin de contaje T01F localizado en el registro INTCON.

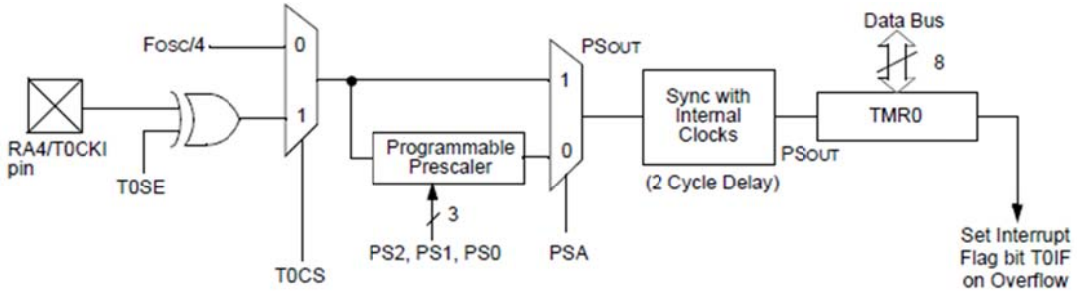
Los impulsos aplicados al TMR0 pueden provenir de los pulsos aplicados al pin T0CKI o de la señal de reloj interna (Fosc/4), lo que le permite actuar de dos formas diferentes:

- Como contador de los impulsos que le llegan por el pin RA4/T0CKI.
- Como temporizador de tiempos.

El actuar de una forma u otra forma depende del bit T0CS del registro OPTION:

- Si T0CS = 1, el TMR0 actúa como contador.
- Si T0CS = 0, el TMR0 actúa como temporizador.

El TMR0 es un registro de propósito especial ubicado en la posición 1 del área SFR de la RAM de datos (tabla 2.2). Puede ser leído y escrito al estar conectado directamente al bus de datos.
La figura 2.7 ofrece el esquema de funcionamiento del TMR0. Se puede leer en cualquier momento para conocer el estado de la cuenta. Cuando se escribe un nuevo valor sobre TMR0 para comenzar una nueva temporización, el siguiente incremento del mismo se retrasa durante los dos ciclos de reloj posteriores.



Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 5-2 for detailed block diagram).

Figura 2.8 *Esquema de funcionamiento del timer principal TMR0*

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
01h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

Tabla 2.4 *Registros asociados con el TIMER 0*

2.3.1.9 La Pila (*Stack*)

La pila es una zona de memoria que se encuentra separada tanto de la memoria de programa como de la de datos dentro del microcontrolador (figura 2.8). Su estructura es del tipo LIFO (*Last In First Out*) por lo que el último dato que se guarda es el primero que sale.

El PIC16F84A dispone de una pila con ocho niveles o registros de una longitud de 13 bits cada uno de ellos.

La manera de cargar la pila es a través de la llamada a subrutina con la instrucción *call*, que almacena el contenido del contador de programa (PC) en la posición superior de la pila. Para recuperar el contenido de la pila en el PC, hay que ejecutar la instrucción de retorno de subrutina *return*.

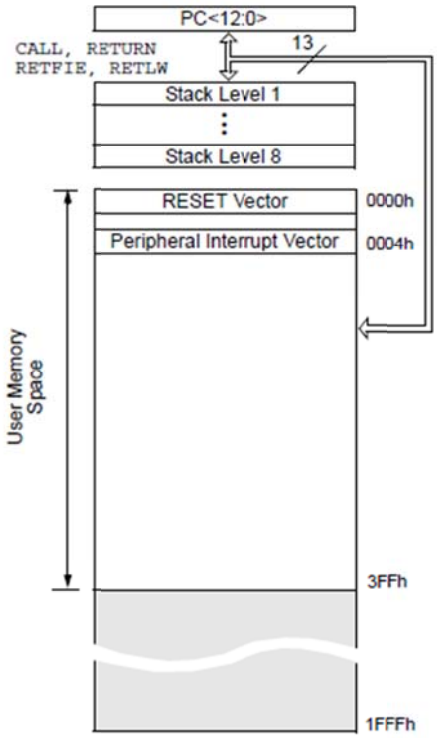


Figura 2.9 *Estructura de la pila y memoria de programa del PIC16F84A*

2.3.1.10 Oscilador

Todo microcontrolador requiere de un circuito que le indique la velocidad de trabajo, es el llamado oscilador o reloj. Este genera una onda cuadrada de alta frecuencia que se utiliza como señal para sincronizar todas las operaciones del sistema. Este circuito es muy simple pero de vital importancia para el buen funcionamiento del sistema. Generalmente todos los componentes del reloj se encuentran integrados en el propio microcontrolador y tan sólo se requieren unos pocos componentes externos, como un cristal de cuarzo o una red RC, para definir la frecuencia de trabajo.

En el PIC16F84A los pines OSC1/CLKIN y OSC2CLKOUT son las líneas utilizadas para este fin. Permite cinco tipos de osciladores para definir la frecuencia de funcionamiento:

- XT: Cristal de cuarzo.
- RC: Oscilador con resistencia y condensador.
- HS: Cristal de alta velocidad
- LP: Cristal para baja frecuencia y bajo consumo de potencia.
- Externa: Cuando se aplica una señal de reloj externa.

Para la realización de este proyecto se ha optado por un oscilador XT debido a su facilidad de montaje y su gran nivel de fiabilidad para generar nuestra frecuencia de reloj deseada, que en este caso es de 4 MHz.

OSCILADOR XT

Está basado en el oscilador de cristal de cuarzo o en un resonador cerámico. Es un oscilador estándar que permite una frecuencia de reloj muy estable comprendida entre 100 KHz y 4 MHz.

En la siguiente figura se muestra la conexión para el PIC16F84A:

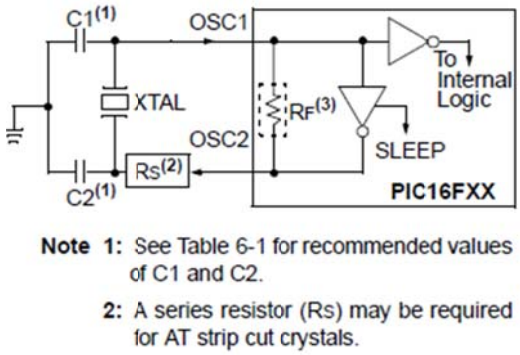


Figura 2.10 *Esquema del oscilador utilizando un cristal de cuarzo*

Capítulo 3. Diseño del Medidor de Distancias por Ultrasonidos

3.1 Software de Desarrollo

Para poder realizar el proyecto se han utilizado dos programas fundamentales, MPLAB y PROTEUS.

El MPLAB consiste en un programa de desarrollo para la programación del PIC16F84A, dicho software permite la realización de programas mediante el lenguaje de programación Ensamblador (*Assembler*), pudiendo compilar y linkar el programa y así obtener el archivo final ya preparado para ser grabado en el microcontrolador mediante un programador externo, en este caso se ha usado el programador GALEP4. El MPALB también permite la simulación de las instrucciones pudiendo así comprobar el correcto funcionamiento del programa a realizar, mediante la simulación de las instrucciones y la información guardada en los registros del microcontrolador.

El PROTEUS es una herramienta de software que permite realizar la simulación de circuitos esquemáticos, cosa que ayuda bastante a la hora de realizar un montaje de pruebas en modo real, ya que si el circuito que estamos diseñando funciona en la simulación, tenemos una gran probabilidad de que funcione con los componentes físicos. Este programa también permite la simulación de microcontroladores, es decir, podemos grabar el programa compilado en MPLAB en el microcontrolador y verificar su funcionamiento. También permite la realización de circuitos impresos por lo que es una herramienta bastante completa a la hora de realizar este proyecto.

3.2 Principios de Diseño para el Medidor de Distancias por Ultrasonidos

La idea principal para realizar el medidor de distancias por ultrasonidos, es aprovechar el pulso digital que nos ofrece el sensor y contabilizar la anchura del mismo mediante el registro TMR0 que dispone nuestro microcontrolador. Siguiendo las características del sensor de ultrasonidos utilizado (figura 1.10), obtenemos la siguiente información:

- Rango de medidas comprendidas entre 3 cm y 3 m (según los datos proporcionados por el fabricante).
- El intervalo del pulso de eco en tiempo va comprendido entre 0 segundos y 36 ms (según los datos proporcionados por el fabricante).

A continuación se muestra el pulso de eco del sensor de ultrasonidos SFR04:

Figura 3.1 *Ancho del pulso de eco del sensor SFR04*

Observando la figura anterior y teniendo en cuenta los datos aportados por el fabricante, se deduce que el sensor comienza a medir a partir de un ancho de pulso a 100 µs y finaliza la medida con el ancho de pulso a 18 ms. Si el sensor no encuentra ningún objeto u obstáculo, el ancho de pulso será de 36 ms. Por lo tanto se deduce la siguiente información:

- Cero metros equivale a 0 segundos
- 3 metros (300 cm) equivale a un ancho de 18 ms

Con lo que aplicando una regla de tres obtenemos lo siguiente:

300 cm → 18 ms
1 cm → X

$$X = \frac{18\ ms\ \cdot\ 1\ cm}{300\ cm}$$

X = 60 µS

Vemos que por cada centímetro medido, el pulso de eco se incrementa 60 µs.

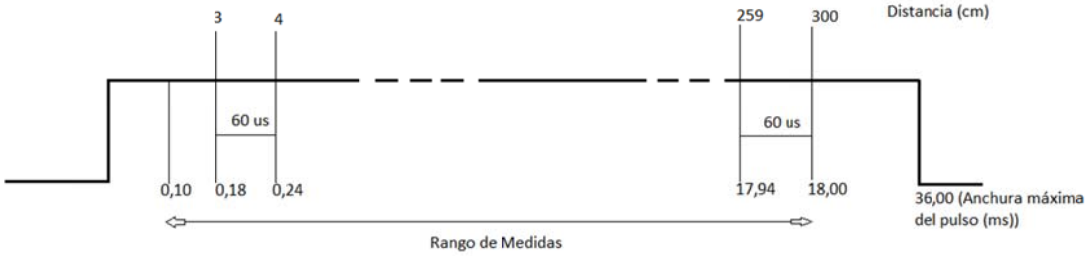


Figura 3.2 *Relación entre la distancia medida y la anchura de pulso*

Debido a que nuestro micontrolador es de 8 bits, nuestra distancia máxima a medir será de 250 cm en vez de 300 cm, ya que el TMR0 acepta un valor en decimal de 256.

3.2.1 Cálculo del Ancho de Pulso mediante el Timer 0

En este apartado se explicará cómo utilizar el Timer 0 para el cálculo de la distancia mediante la medida del ancho de pulso de eco.

Anteriormente, en el apartado 2.3.1.8 se realizó una breve explicación de esta función que dispone el PIC16F84A. A continuación aplicaremos esta función para el objetivo de medida en el proyecto.

Para la realización de la medida del ancho de pulso, activaremos el TMR0 como temporizador. Por lo tanto, cuando el TMR0 trabaja como temporizador cuenta los impulsos de la señal interna de reloj **Fosc/4** (véase figura 2.8). Estos impulsos tienen una duración conocida de un ciclo máquina, que es la unidad básica de tiempo que utiliza el microcontrolador. Un ciclo máquina equivale a cuatro ciclos de la señal de reloj. Para calcular el tiempo de un ciclo máquina utilizamos la siguiente formula:

$$\boldsymbol{Tiempo = 4\frac{1}{f}CM} \qquad \textit{(3.1)}$$

Siendo:

- f, la frecuencia de oscilación (en nuestro caso 4 MHz).
- CM, ciclos máquina que tarda en ejecutar una tarea (en este caso 1).

Al utilizar un cristal de 4 MHz para la realización del oscilador, obtenemos una señal de reloj de 4 MHz, por lo que la duración de cada ciclo máquina será de 1 µs. Por lo tanto los impulsos del TMR0 se incrementaran cada ese periodo de tiempo (1 µs).

Como el TMR0 es un contador ascendente, debe ser cargado con el valor de los impulsos que se desean contar restados de 256 (b'11111111' 8 bits), que es el valor de desbordamiento.

Para el cálculo del desbordamiento a un tiempo deseado, utilizaremos la siguiente formula:

$$\boldsymbol{Temporización = TCM \cdot Prescaler \cdot (256 - Carga\,TMR0)} \qquad \textit{(3.2)}$$

$$\boldsymbol{CargaTMR0 = 256 - \frac{Temporización}{TCM \cdot Prescaler}} \qquad \textit{(3.3)}$$

Donde:

- Temporización, el tiempo que queremos contar (en nuestro caso 60 µs).
- TCM, tiempo de ciclo máquina (en nuestro caso 1 µs).
- Prescaler, divisor de frecuencia que utilizamos para el TMR0, cuya configuración se habilita mediante cuatro flags ubicados en el registro OPTION_REG (PSA, PS2, PS1 y PS0). En nuestro caso la configuración está diseñada de la siguiente manera:
    1. PS0 = 0. El divisor de frecuencia se asigna al TMR0.
    2. PS2, PS1 y PS0 = 0 0 0. El divisor del TMR0 es 1:2, en nuestro caso no es necesario un divisor de frecuencia ya que la temporización que se necesita es pequeña. Pero la configuración de estos tres bits nos obliga a poner un divisor de frecuencia por dos.
- CargaTMR0, valor que nos interesa calcular para cargar el Timer 0 y así provocar el desbordamiento y por consiguiente obtener el conteo.

3.3 Estructura del Patillaje del PIC16F84A

Para la realización del proyecto y posterior realización del código fuente, primero hemos de esquematizar como configuraremos el microcontrolador PIC16F84A para la realización del medidor de distancias por ultrasonidos. Teniendo que utilizar una de las entradas del PIC para activar el sensor y otra entrada para medir el pulso de eco. De las cuatro entradas que dispone el PIC (RA4:RA0), se puede elegir cualquiera, menos la RA4, para utilizarla como señal de activación. Para la señal de eco se tiene que utilizar obligatoriamente el pin de entrada RA4, ya que en este pin se encuentra la aplicación del Timer 0 explicado anteriormente y que más adelante explicaremos con más detalle para el cálculo del ancho de pulso.

Como nuestra finalidad es visualizar la distancia en un LCD, hemos de tener en cuenta que para el control del LCD será necesario utilizar pines del PIC.
Quedando inicialmente el siguiente esquema teórico:

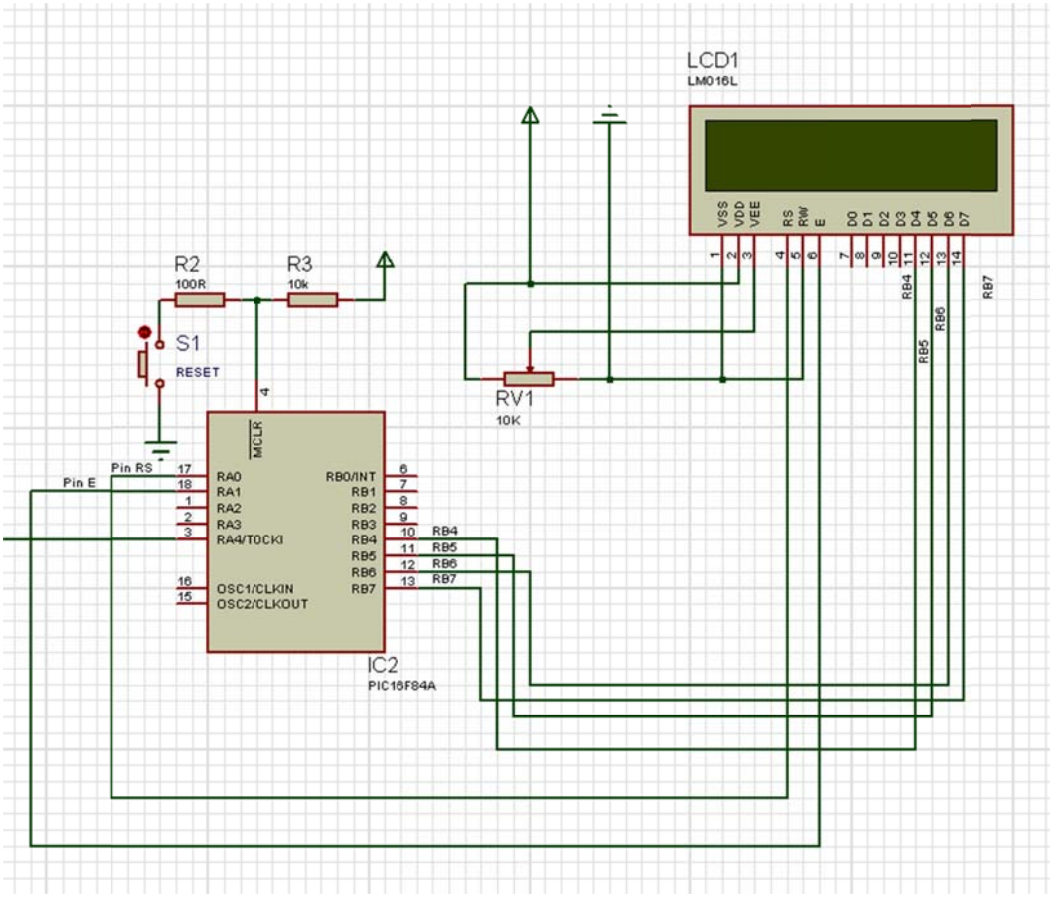


Figura 3.3 *Esquema teórico inicial para la medida de distancias*

En la figura anterior se puede observar que utilizamos los pines RA0 y RA1 para el control del LCD, el pin RA3 para activar el sensor y el pin RA4 para medir el ancho del pulso de eco.
Las salidas RB7:RB4 las utilizaremos como bus de comunicaciones con el módulo LCD para visualizar la distancia. Las resistencias mostradas en el circuito únicamente sirven para realizar la función de reset al PIC.

Se puede observar que en los pines 15 y 16 no hay nada conectado, ya que el PROTEUS permite configurar la frecuencia de oscilación del PIC a la que deseemos.

3.4 Estructura del Software Inicial

En este apartado se detalla la estructura de código fuente que se realizó inicialmente para el medidor de distancias por ultrasonidos.

3.4.1 Inicio del Programa y Subrutinas de Eco

A continuación se muestra la estructura del código fuente correspondiente a los módulos de “Inicio”, “Principal” y a los módulos encargados de la medida del pulso de eco:

```
Inicio*
        call    LCD_Inicializa
        bsf     STATUS,RP0
        bcf     Disparo
        bsf     Eco
        movlw   b'00000000'
        movwf   OPTION_REG
        bcf     STATUS,RP0
        bcf     Disparo
Principal
        clrf    Distancia
        bsf     Disparo
        call    Retardo_20micros
        bcf     Disparo
Espera_Inicio_Eco
        btfss   Eco
        goto    Espera_Inicio_Eco
        movlw   TMR0_Carga60micros
        movwf   TMR0
        movlw   b'10100000'
        movwf   INTCON
Espera_Final_Eco
        btfsc   Eco
        goto    Espera_Final_Eco
        clrf    INTCON
        call    Visualiza
        call    Retardo_2s
Fin     goto    Principal
```

- *Estructura del programa inicial (que posteriormente es modificado).
- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

En esta estructura de código ejecutamos la orden de disparo para que se active el sensor. Al comienzo inicializamos el LCD, configuramos la entrada del PIC que se encarga de emitir el pulso de disparo como salida e inicializamos a 0 el registro que guardará el valor de la distancia (registro "Distancia", ubicado en las direcciones libres de la RAM). Observando las características del fabricante (figura 1.10) podemos ver que el pulso de disparo necesita una duración de como mínimo 10 µs, de ahí que utilicemos un retardo de 20 µs para generar este pulso.
Las siguientes instrucciones se encargan de contabilizar el pulso de eco mediante la espera del flanco de subida y del flanco de bajada. Cuando el sensor genere el flanco de subida el programa automáticamente generará la interrupción, la interrupción se encargará de contabilizar la duración del pulso por medio del TMR0, más adelante se detallará con más detalle la subrutina de interrupción.

Para verificar el funcionamiento de esta parte del programa se ha utilizado las herramientas de simulación del MPLAB. De esta manera nos asegurábamos que hasta que no detectara un flanco de subida y otro de bajada, el programa no continuara.
Para realizar esta simulación se ha usado el siguiente cuadro de configuración:

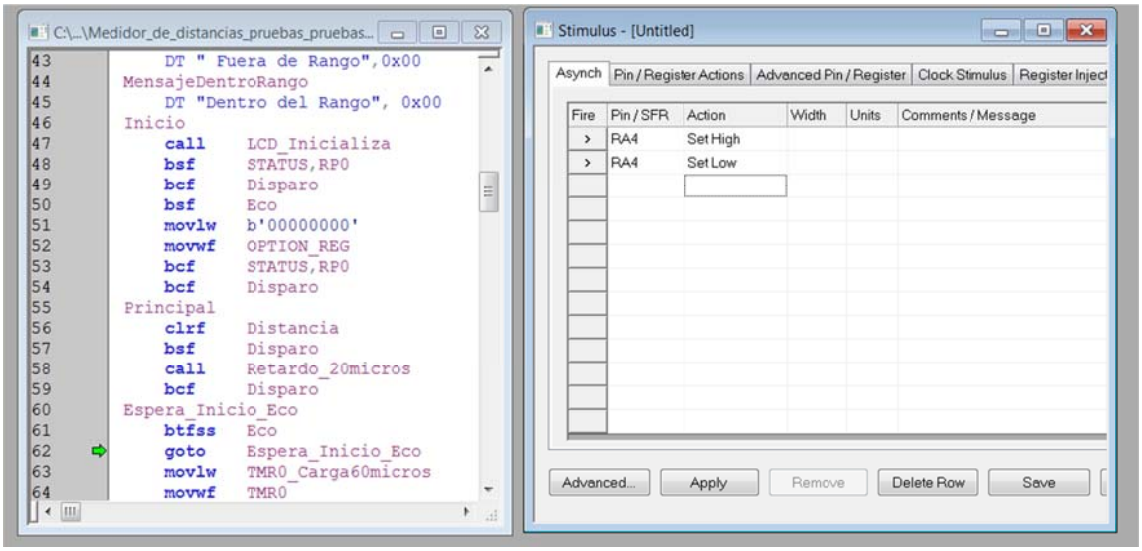


Figura 3.4 *Visualización en MPLAB de la simulación del flanco de subida y bajada del pulso de eco*

3.4.2 Estudio de la Subrutina de Interrupción para el Cálculo del Pulso de Eco

A continuación se muestra el código fuente correspondiente a la subrutina de interrupción, encargada de calcular el ancho de pulso de eco.

```
Interrupcion_Timer0*
        movlw  TMR0_Carga60micros
        movwf  TMR0
        movlw  .1
        addwf   Distancia,F
        movlw  MAXIMA_DISTANCIA
        btfsc    STATUS,C
        movwf  Distancia
Final_de_la_Interrupcion
        bcf      INTCON,T0IF
        retfie
```

- *Estructura del programa inicial (que posteriormente es modificado).
- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

Inicialmente en el programa se le asigna a la variable “TMR0_Carga60micros” un valor decimal de 226 (TMR0_Carga55micros EQU .226) que es el resultado de restar 256 menos 25. El 256 es el tiempo máximo del timer 0 y el 25 es el tiempo obtenido para que haga una interrupción cada 60 µs (resultado obtenido de la fórmula 3.3).

Esta parte del código se encarga de contar pulsos de 55 µs, por cada pulso obtenido se va incrementando en 1 el valor del registro “Distancia” y compara dicho registro con la variable “MAXIMA_DISTANCIA”, configurada inicialmente con un valor decimal de 250. Si la distancia obtenida es superior a 250, el registro de trabajo W almacenará el valor de la distancia máxima si no almacenará el valor guardado en el registro “Distancia”. Las dos últimas instrucciones son las encargadas de inicializar el TMR0 y retornar al programa principal.

3.4.3 Subrutinas para el Cálculo de la Distancia

Código fuente encargado del cálculo de la distancia:

```
Visualiza*
        call    LCD_Borra
        movlw   MINIMA_DISTANCIA
        subwf   Distancia,W
        btfss   STATUS,C
        goto    DistanciaMenor
        movf    Distancia,W
        sublw   MAXIMA_DISTANCIA
        btfsc   STATUS,C
        goto    DistanciaFiable
DistanciaMayor
        movlw   MensajeFueraRango
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MAXIMA_DISTANCIA
        movwf   Distancia
        movlw   MensajeDistanciaMayor
        goto    VisualizaDistancia
DistanciaMenor
        movlw   MensajeFueraRango
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MINIMA_DISTANCIA
        movwf   Distancia
        movlw   MensajeDistanciaMenor
        goto    VisualizaDistancia
DistanciaFiable
        movlw   MensajeDentroRango
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MensajeDistancia
```

- *Estructura del programa inicial (que posteriormente es modificado).
- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

Estas líneas de código se encargan de realizar todos los cálculos pertinentes para determinar si la distancia medida por el sensor está dentro del rango definido o no. Lo que hacemos en estas líneas es comparar el valor almacenado en el registro "Distancia" con los valores de "MAXIMA_DISTANCIA" (250 cm) y "MÍNIMA_DISTANCIA" (3 cm). Dependiendo del resultado de la comparación, el módulo LCD mostrará los siguientes mensajes:

- DISTANCIA FUERA DEL RANGO – DIST. MENOR DE 3 CM
- DISTANCIA FUERA DEL RANGO – DIST. MAYOR DE 250 CM
- DISTANCIA DENTRO DEL RANGO – DISTANCIA 'X' CM

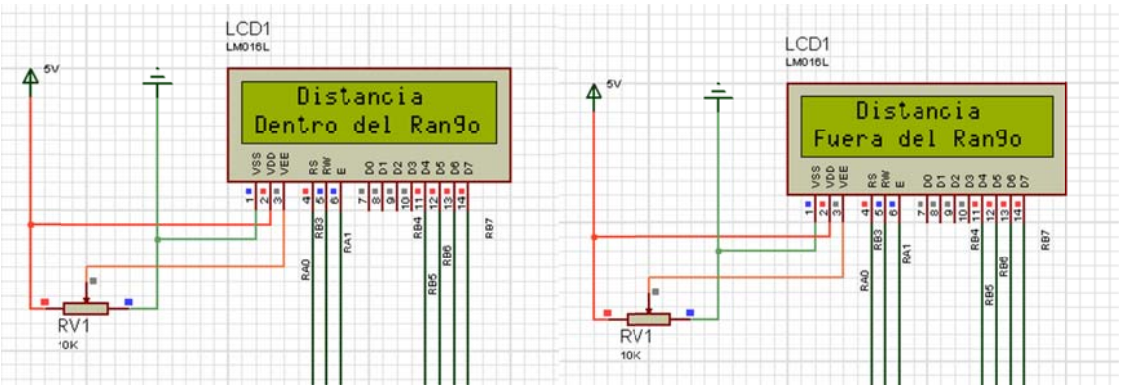


Figura 3.5 *Demostración de la visualización de los mensajes en el LCD*

3.4.4 Subrutinas para la Visualización de la Distancia

Código fuente encargado de la visualización de la distancia:

```
VisualizaDistancia*
        call    LCD_Mensaje
        movlw   .5
        call    LCD_PosicionLinea2
        movf    Distancia,W
        call    BIN_a_BCD
        movf    BCD_Centenas,W
        btfss   STATUS,Z
        goto    VisualizaCentenas
        movf    Distancia,W
        call    BIN_a_BCD
        call    LCD_Byte
        goto    Visualiza_cm
VisualizaCentenas
        call    LCD_Nibble
```

```
        movf    Distancia,W
        call    BIN_a_BCD
        call    LCD_ByteCompleto
Visualiza_cm
        movlw   MensajeCentimetro
        call    LCD_Mensaje
```

- *Estructura del programa inicial (que posteriormente es modificado).
- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

Estas líneas de código se encargan de convertir la información almacenada en el registro "Distancia" a formato BCD. Ya que la información está guardada en formato binario y para poder ser mostrada en el módulo LCD tiene que ser convertida en formato BCD. Para ello se utiliza una librería llamada "BIN_BCD_Medidor_Distancias_por_Ultrasonidos.INC" que se encarga de realizar esta conversión. Las líneas de código que forman esta librería están detalladas en el anexo.

3.4.5 Verificación del Incremento de la Distancia mediante Software

A continuación, mediante un ejemplo, se muestra como se ha verificado el correcto funcionamiento de nuestro código fuente a través de los programas MPLAB y PROTEUS.

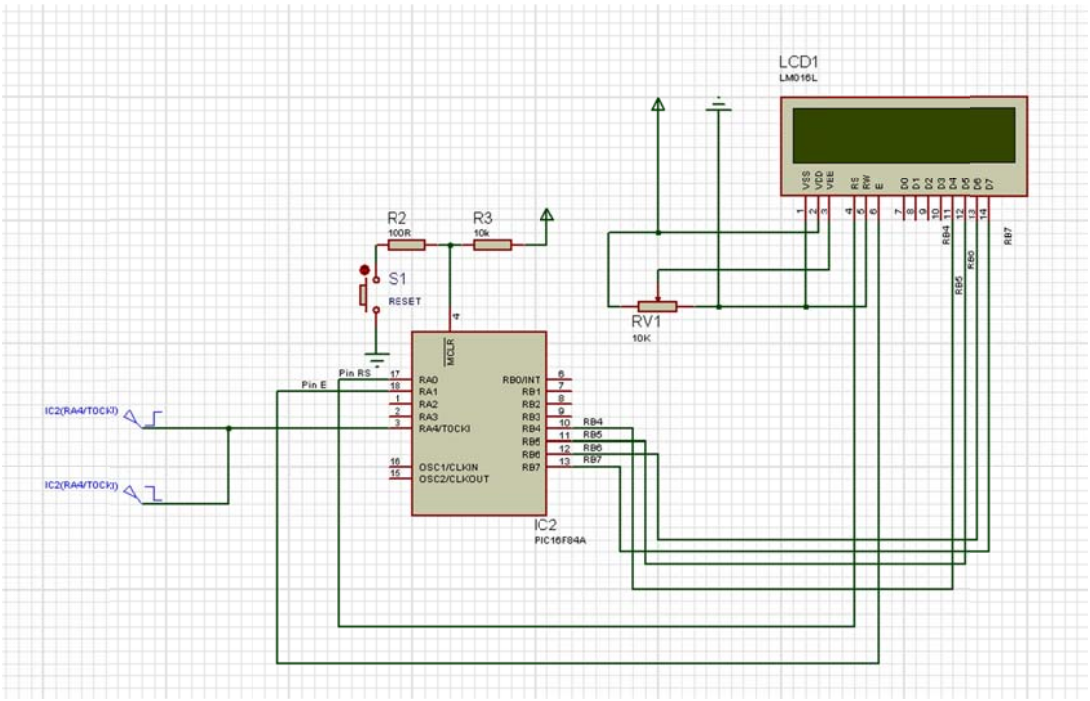


Figura 3.6 *Circuito esquemático realizado en PROTEUS para simular el pulso de eco*

El circuito de la figura 3.5 está formado por el PIC16F84A, el módulo LCD, dos resistencias conectadas a un pulsador para generar un "Reset" al microcontrolador (en el caso que hiciera falta) y un potenciómetro (RV1) que únicamente sirve para ajustar el contraste del LCD. Dicho circuito sirve para simular el pulso de eco, para realizar la simulación simplemente se tiene que generar dos flancos, uno de subida y otro de bajada, y colocarlos en el pin RA4 del PIC. Para averiguar el tiempo al que se tiene que configurar el flanco de subida hay que utilizar una opción del MPLAB que nos permite visualizar el tiempo de ejecución para cada instrucción, en la siguiente figura se muestra el tiempo que tarda el programa en llegar a la estructura de "Espera_Inicio_Eco".

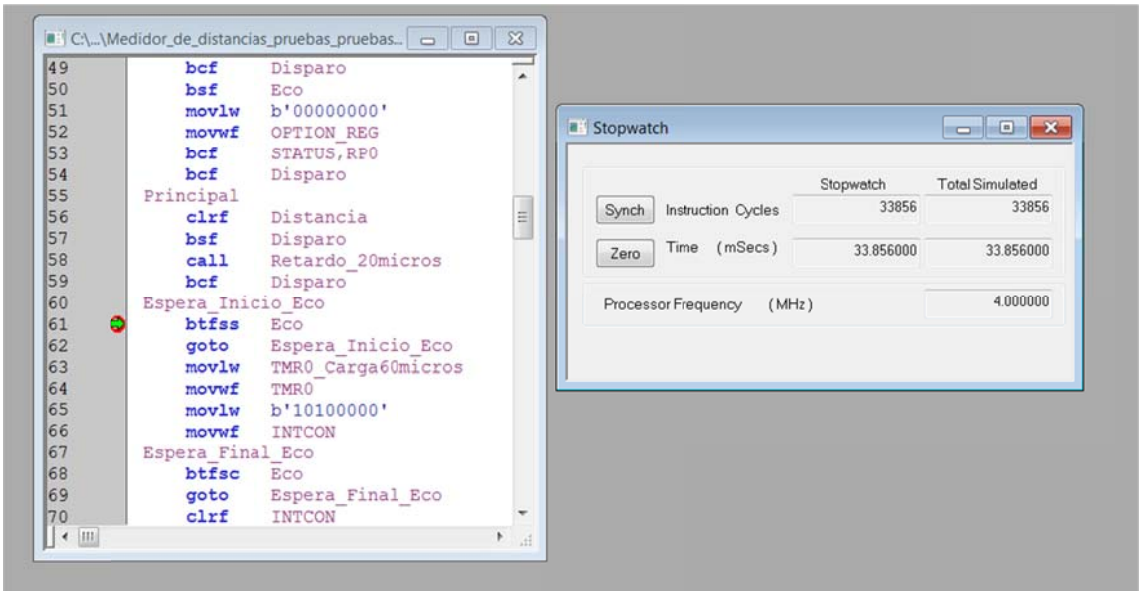


Figura 3.7 *Visualización de la duración del tiempo en llegar a "Espera_Inicio_Eco"*

Por lo tanto el tiempo de inicio de eco para configurar en PROTEUS será de 33 ms. Previamente habiendo configurado en MPLAB la frecuencia de oscilación del PIC a 4 MHz.
Cada incremento de tiempo en el flanco de bajada servirá para verificar el funcionamiento del programa a la hora de calcular la distancia, ya que por cada incremento de 60 µs (aprox.) el valor de la distancia deberá incrementarse 1 cm respectivamente.

A continuación se muestran dos figuras que simulan el ejemplo de incremento de la distancia:

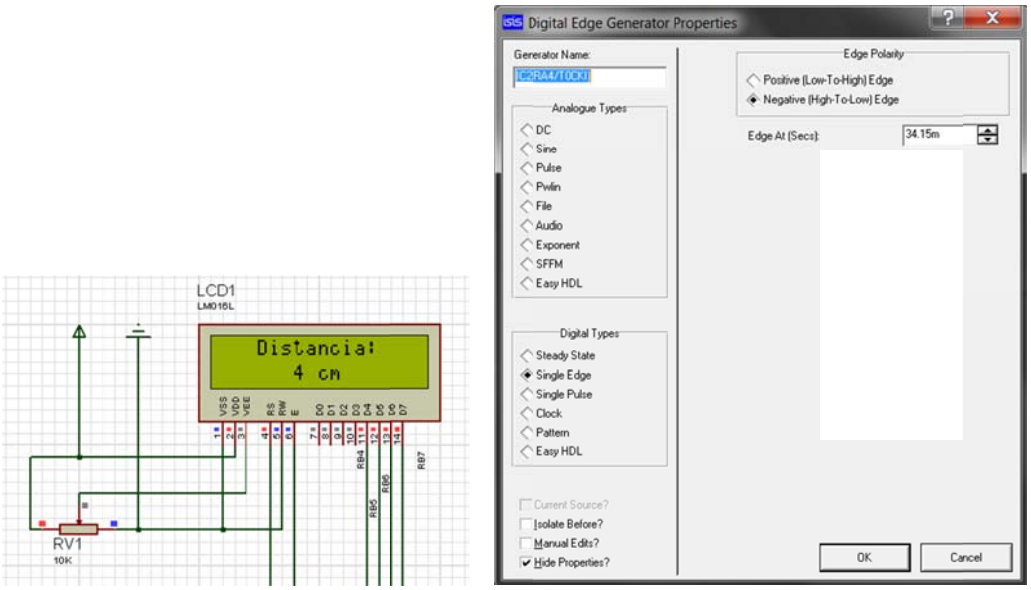


Figura 3.8 *Configuración del flanco de bajada para simular una distancia*

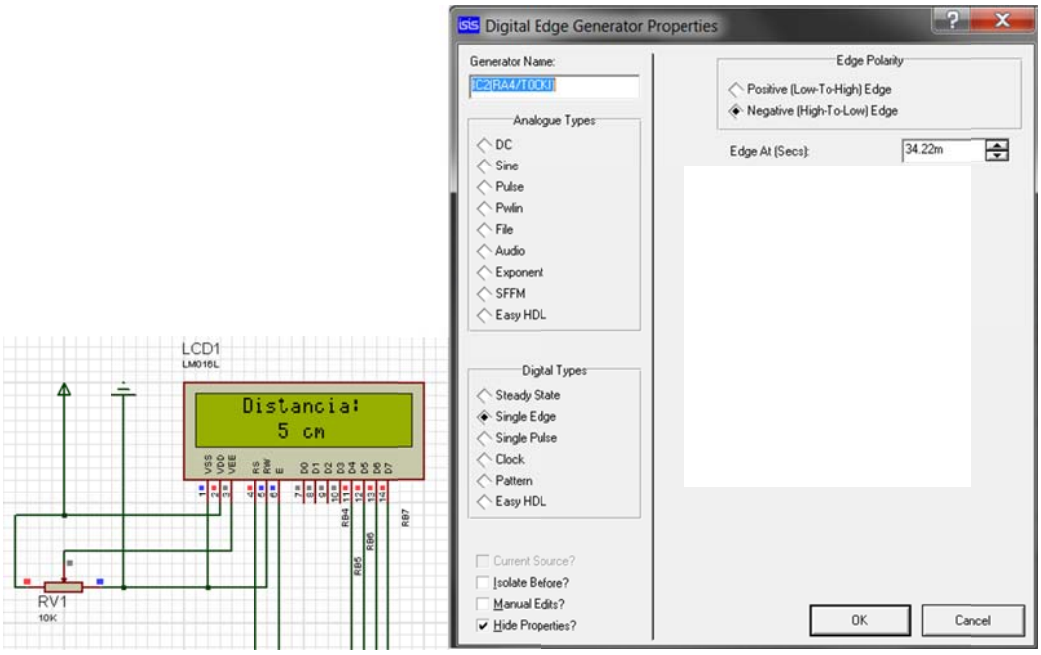


Figura 3.9 *Configuración del flanco de bajada para incrementar la distancia anterior mediante simulación*

3.5 Modificaciones en la Estructura del Programa

Una vez comprobado que el programa funciona correctamente, se pueden realizar las modificaciones adecuadas en el software para que el proyecto sea mucho más óptimo.

Mejoras realizadas:

- **Pulsador de disparo:** Con esta modificación el usuario podrá decidir cuando realizar la medida.
- **Visualización de mensajes:** Mostrar mensajes más precisos en el módulo LCD, para que el usuario tenga una mejor compresión de la información mostrada por el dispositivo.

3.5.1 Estructura de la Activación de Disparo

A continuación se muestra el siguiente código fuente:

```
Activacion_Disparo
        btfsc   PulsadorActivacion
        goto    Activacion_Disparo
        call    Retardo_20ms
        btfsc   PulsadorActivacion
        goto    Activacion_Disparo
        call    LCD_Borra
        movlw   MensajeMidiendo
        call    LCD_Mensaje
        call    Retardo_2s
        bsf     Disparo
        call    Retardo_20micros
        bcf     Disparo
```

- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

Estas líneas de código se encargan de la ejecución de la orden de disparar, es decir, si no pusiéramos estas líneas nuestro sensor estaría constantemente midiendo. Por lo tanto esta parte del código controla el disparo, hasta que el usuario no de la orden de medir, el programa no activará el sensor. En la siguiente figura vemos el esquema teórico del circuito que realiza esta acción:

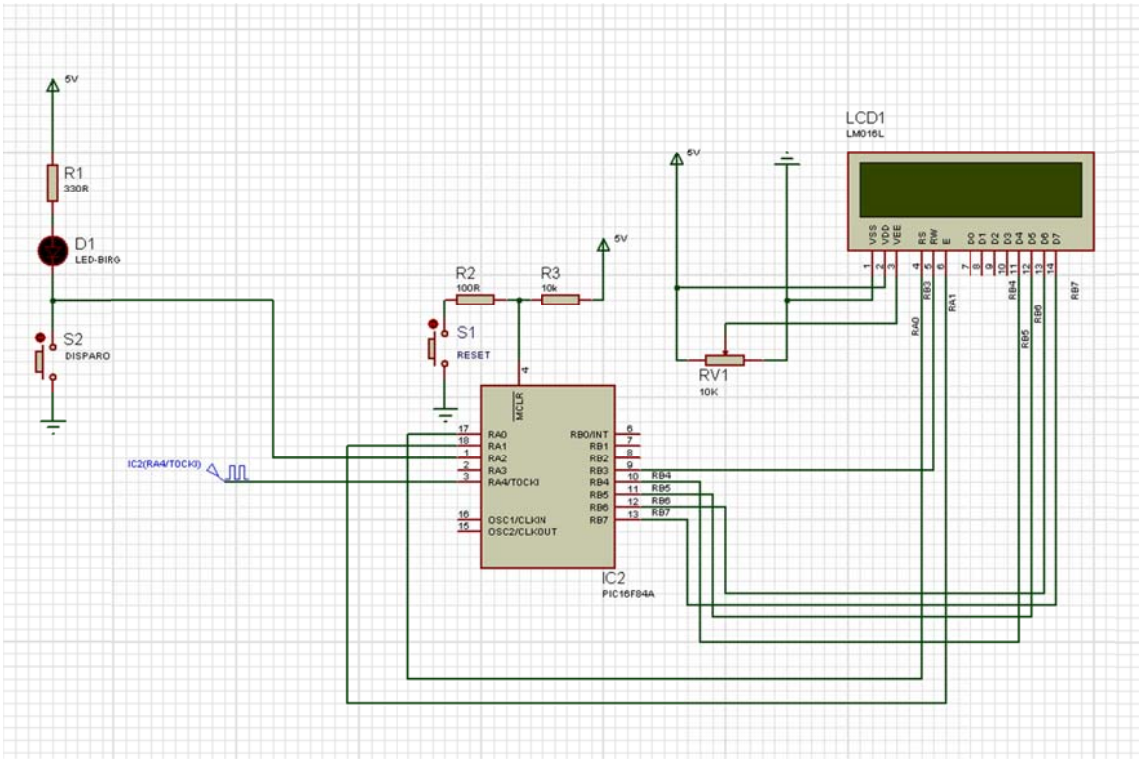


Figura 3.10 *Esquema teórico con el pulsador de disparo*

En la figura podemos observar como se ha utilizado el pin RA2 como activación de disparo, conectado a dicho pin un pulsador. Cada vez que se active este pulsador le enviará un 0 a la entrada del PIC y éste seguidamente generará el pulso de disparo para activar el sensor de ultrasonidos.

En este circuito ya utilizamos en la entrada del pin RA4 una señal de reloj que nos servirá para simular el pulso de eco, independientemente del valor de distancia que nos dé, ya que anteriormente habíamos verificado que el programa realizaba correctamente el incremento de la distancia.

3.5.2 Estructura de los Mensajes

A continuación se muestra todo el código fuente relacionado con este apartado:

```
Mensajes
        addwf   PCL,F
MensajeDistancia
        DT "Dist.:  ",          0x00
MensajeDistancia2
        DT "   Distancia",      0x00
MensajeCentimetro
        DT " cm",               0x00
MensajeDistanciaMenor
        DT "Dist. Menor de:",   0x00
MensajeDistanciaMenor2
        DT "     4 cm",         0x00
MensajeDistanciaMayor
        DT "Dist. Mayor de:",   0x00
MensajeDistanciaMayor2
        DT "     250 cm",       0x00
MensajeFueraRango
        DT "Fuera del Rango",   0x00
MensajeDentroRango
        DT "Dentro del Rango", 0x00
MensajeInicioMedida
        DT "  Para Medir",      0x00
MensajeMidiendo
        DT "  Midiendo...",     0x00
MensajeDisparo
        DT " Pulsar Disparo",   0x00
MensajePulsarParaContinuar
        DT "Pulsar Disparo",    0x00
```

- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

Estas líneas de código sirven para describir el texto que se quiere mostrar en el módulo LCD.

3.5.3 Subrutina de Finalización del Programa

Código fuente que forma esta subrutina:

```
Activacion_Disparo_Para_Volver_a_medir
        btfsc   PulsadorActivacion
        goto    Activacion_Disparo_Para_Volver_a_medir
        call    Retardo_20ms
        btfsc   PulsadorActivacion
        goto     Activacion_Disparo_Para_Volver_a_medir
        call    LCD_Borra
        return
```

- En el Anexo I (pág.91) se puede encontrar la estructura completa del programa final (Flujograma incluido) y una explicación detallada de cada línea de código.

Estas líneas se encargan de mantener la visualización permanentemente en el módulo LCD hasta que el usuario no vuelva a pulsar el botón de "Disparo". A continuación se muestra un ejemplo de lo que aparece en el módulo LCD:

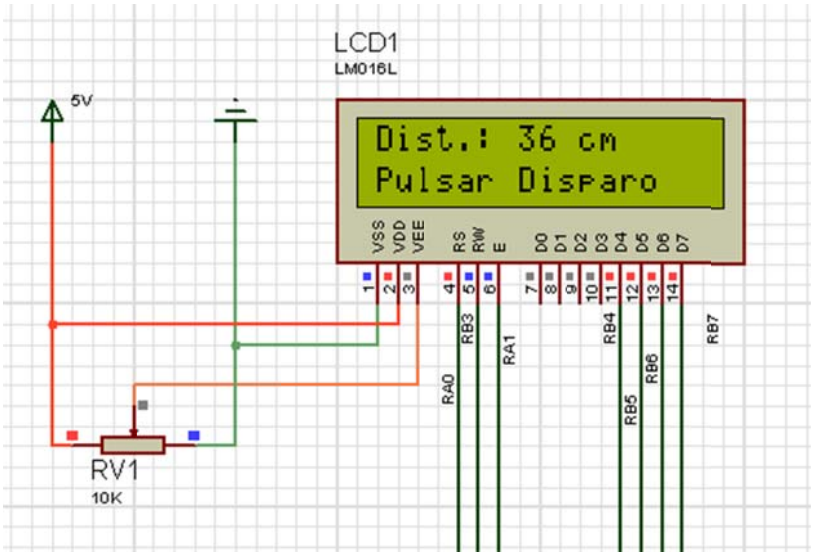


Figura 3.11 *Demostración de la visualización*

3.6 Librerías necesarias para el Correcto Funcionamiento del Programa

En las líneas del programa principal se encuentran instrucciones cuya función es llamar (*call*) a subrutinas no ubicadas en la estructura principal, estas subrutinas se encuentran en librerías, ya que así ahorramos líneas de código en el programa principal con lo que a su vez ahorramos espacio en la memoria del PIC y optimizamos el rendimiento del programa.

Las librerías utilizadas en el programa son:

- P16F84A.INC
- RETARDOS_Medidor_Distancias_por_Ultrasonidos.INC
- LCD_4BIT_Medidor_Distancias_por_Ultrasonidos.INC
- LCD_MENSAJES_Medidor_Distancias_por_Ultrasonidos.INC
- BIN_BCD_Medidor_Distancias_por_Ultrasonidos.INC

P16F84A.INC

En esta librería se localizan las etiquetas que nombran a los diferentes registros y el valor que le corresponde a cada uno. Es decir, se muestra como hay que nombrar a todos los registros propios del microcontrolador. Esta librería se localiza en el directorio principal del programa ensamblador y su contenido se muestra en el anexo.

RETARDOS_Medidor_Distancias_por_Ultrasonidos.INC

Para aplicaciones con microcontroladores resulta necesario generar tiempos de espera, denominados tiempos de retardo. Las instrucciones en el microcontrolador PIC16F84A necesitan un ciclo máquina (4 ciclos de reloj) para ejecutarse, excepto las instrucciones de salto (goto, call, return, etc.) que necesitan dos ciclos máquinas.
En esta librería se encuentran diferentes subrutinas con diferentes cálculos de tiempos de retardo, cuyos tiempos utilizaremos en el programa principal.

LCD_4BIT_Medidor_Distancias_por_Ultrasonidos.INC

En esta librería encontramos todas las subrutinas que permiten realizar las tareas básicas de control de un módulo LCD de 2 líneas por 16 caracteres. , pudiendo así acceder continuamente cuando se necesite para ejecutar subrutinas como LCD_BORRA, LCD_PosicionLinea1, LCD_PosicionLinea2, etc.

LCD_MENSAJES_Medidor_Distancias_por_Ultrasonidos.INC

Esta librería tiene las subrutinas encargadas para el manejo de mensajes a visualizar en un módulo LCD.

BIN_BCD_Medidor_Distancias_por_Ultrasonidos.INC

En esta librería encontramos el programa que convertirá un número binario natural a formato BCD

Todas las librerías tienen la estructura de un fichero (*.INC) y tienen que ser guardadas en la misma carpeta donde se encuentra el fichero del programa principal (*.asm). En el programa principal quedan estructuradas de la siguiente manera:

```
INCLUDE  <P16F84A.INC>
INCLUDE  <RETARDOS_Medidor_Distancias_por_Ultrasonidos.INC>
INCLUDE  <LCD_4BIT_Medidor_Distancias_por_Ultrasonidos.INC>
INCLUDE  <LCD_MENSAJES_Medidor_Distancias_por_Ultrasonidos.INC>
INCLUDE  <BIN_BCD_Medidor_Distancias_por_Ultrasonidos.INC>
```

- En el Anexo I (pág.97 hasta pág.106) se encuentra detallado y explicado todo el código de cada una de las librerías.

Capítulo 4. Realización y Verificación del Hardware

4.1 Placa de Pruebas

Una vez que ha sido verificado el software mediante las simulaciones realizadas por el programa MPLAB y PROTEUS, se pasó a construir una placa de pruebas para comprobar el funcionamiento de todo el hardware diseñado en el PROTEUS.

A continuación se muestra el esquema teórico final del proyecto:

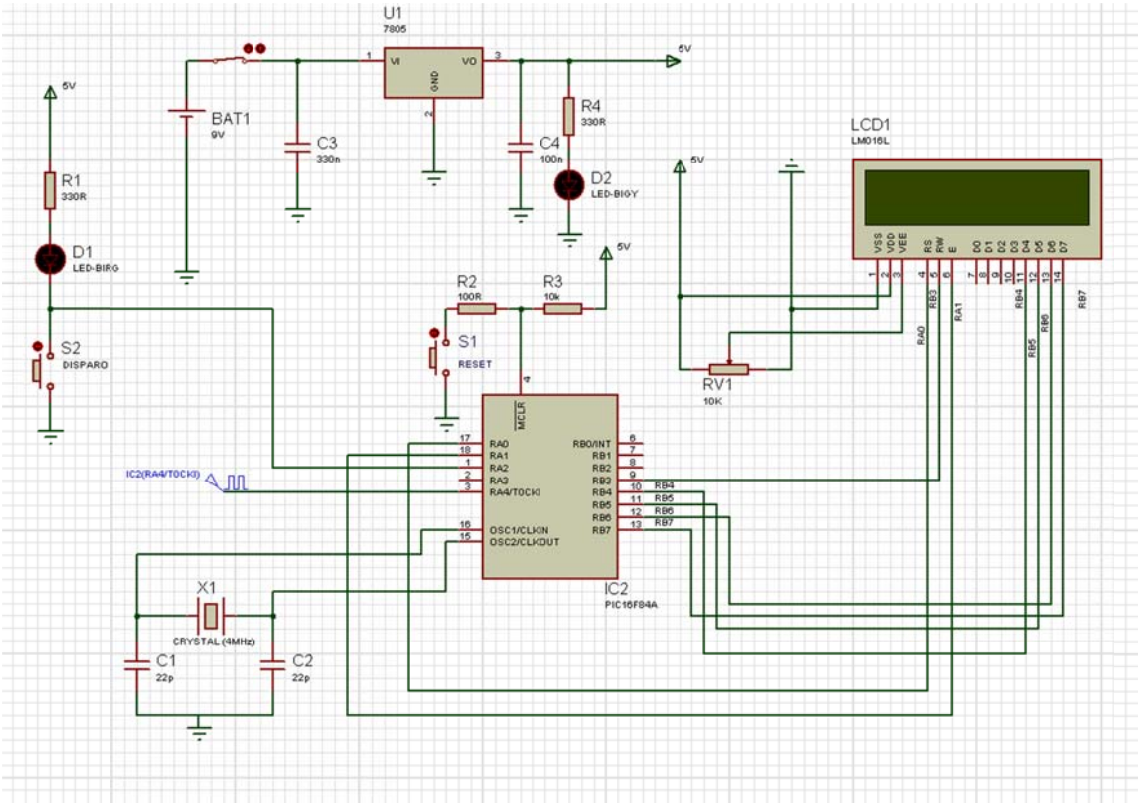


Figura 4.1 *Esquema teórico final sin ejecutar la simulación*

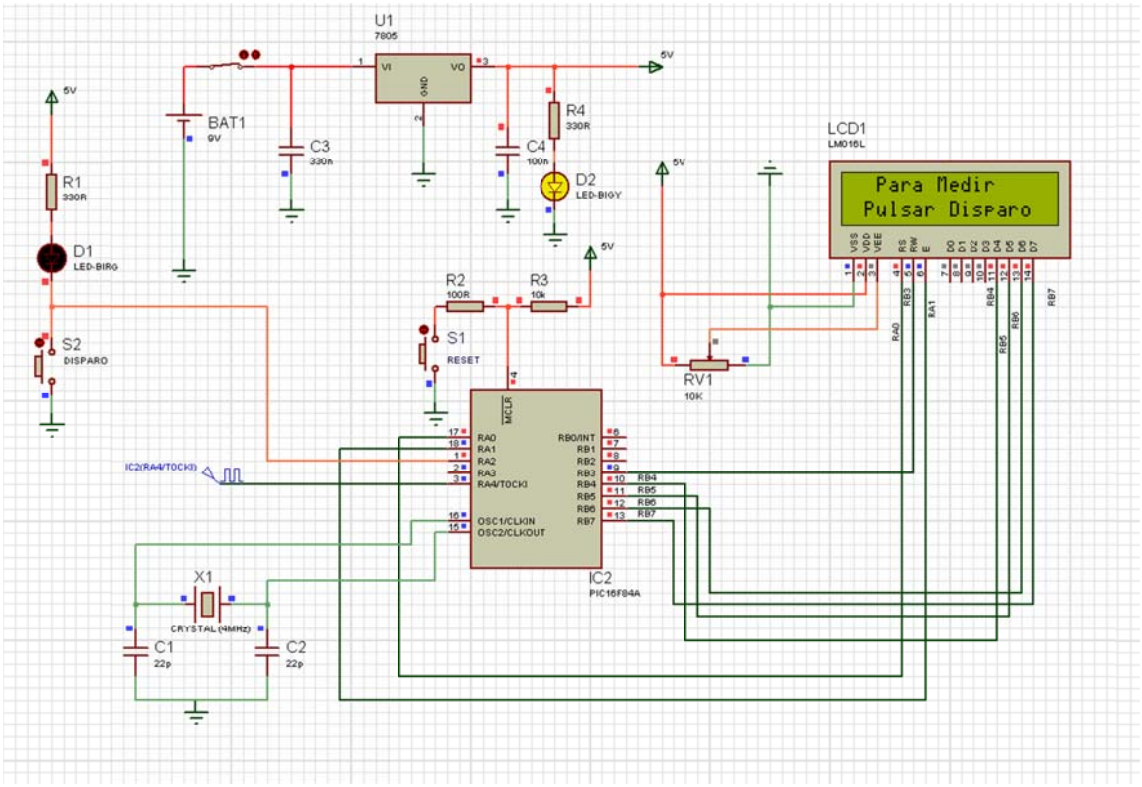


Figura 4.2 *Esquema teórico final ejecutando la simulación*

En la figura podemos observar que se ha añadido el circuito correspondiente al oscilador, formado por dos condensadores de 22pF y un cristal de cuarzo de 4 MHz. Para realizar las simulaciones el circuito no afectaba en el funcionamiento, pero a la hora de hacer el montaje físico es necesario utilizarlo. También se puede observar que hemos añadido un circuito regulador de tensión, cuya función es convertir una tensión de 9V a 5V. Este circuito es necesario ya que el objetivo es hacer un medidor de distancias por ultrasonidos portable, es decir, que podamos transportar el medidor de un lado a otro. Por este motivo se necesita un circuito que convierta los 9V de la pila a 5V, que es la tensión a la que está alimentado tanto el microcontrolador como el LCD.

A continuación se muestran unas imágenes de la placa de pruebas fabricada:

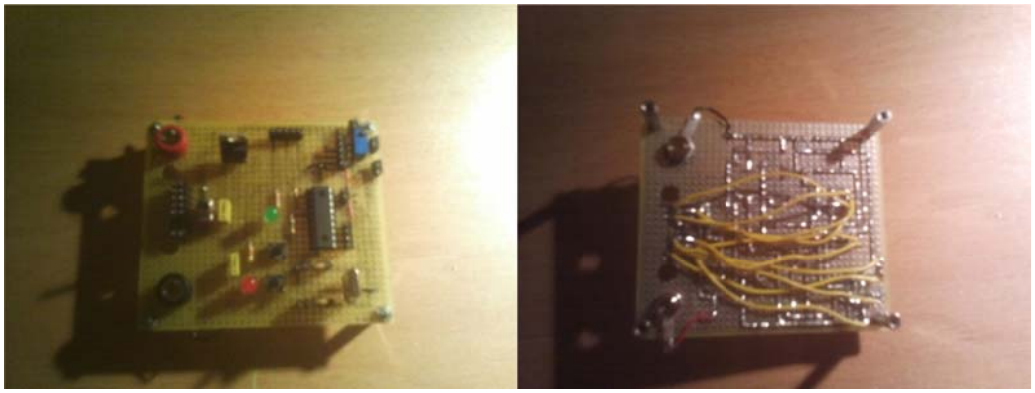


Figura 4.3 *Vista de la cara de componentes y la cara de soldaduras*

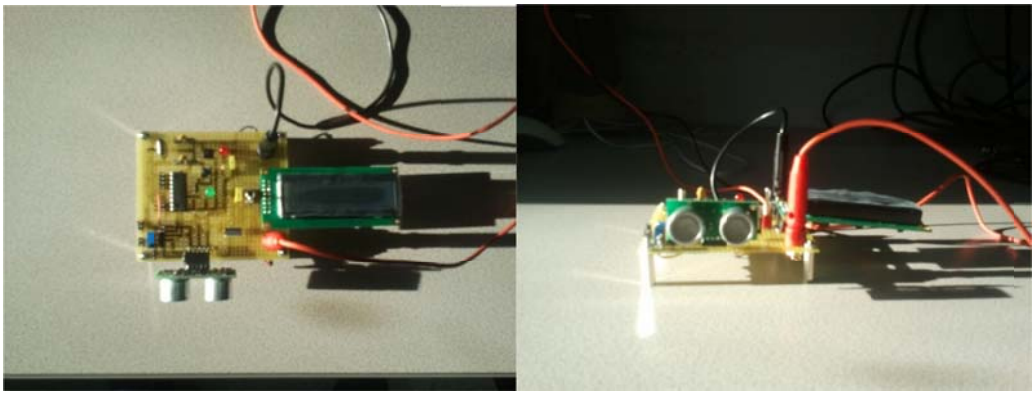


Figura 4.4 *Vista superior y frontal de la placa con la incorporación del sensor y del módulo LCD*

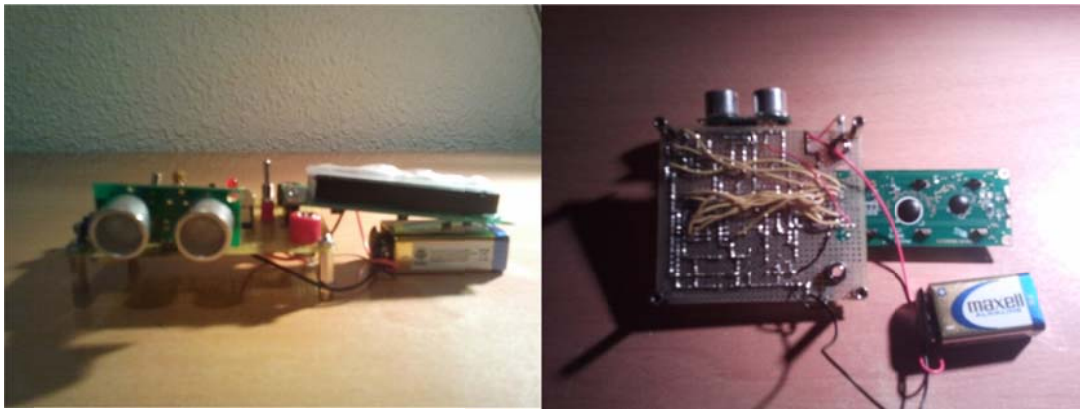


Figura 4.5 *Vista frontal e inferior de la placa con la incorporación de la pila*

La función principal de la placa de pruebas es verificar el funcionamiento diseñado por software, aparte de para realizar las modificaciones necesarias en cuestión de software y mejoras. También se fabricó con la idea de obtener medidas visuales y gráficas mediante el osciloscopio. Por eso está diseñada con varios tira-pines hembra y con dos bornes para la alimentación por fuente de alimentación.

En las siguientes imágenes se muestran un par de gráficas por osciloscopio donde podemos observar el comportamiento físico del sensor de ultrasonidos:

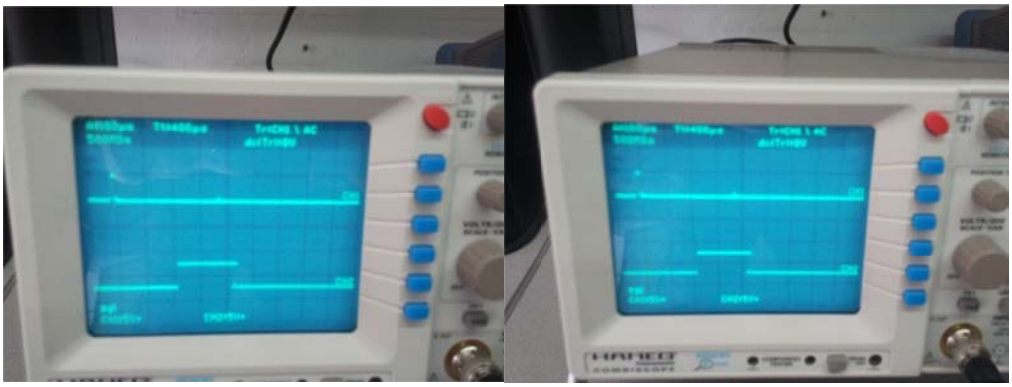


Figura 4.6 *Comportamiento del sensor de ultrasonidos*

En la figura se observa el pulso de disparo (Canal 1 del osciloscopio) y el pulso de eco (Canal 2 del osciloscopio) a la hora de realizar una medida. Cada vez que la medida es de mayor distancia, este pulso de eco irá incrementando proporcionalmente su longitud.

4.2 Modificaciones y Mejoras

4.2.1 Modificación de la Distancia Mínima

Una modificación realizada ha sido la rectificación de la distancia mínima. A la hora de realizar pruebas para medir distancias, observamos que al medir una distancia inferior o igual a 3 cm el sensor se comporta correctamente, mostrando el mensaje de "Distancia menor a 3 cm" por el módulo LCD. El problema se encuentra en el momento que se mide una distancia de 4 cm, ya que el módulo LCD vuelve a mostrar el mismo mensaje "Distancia menor a 3 cm". Con la ayuda del osciloscopio se consigue averiguar el problema, ya que se apreciaba que la longitud del pulso de eco no incrementaba nada para esta distancia, es decir, las medidas comprendidas entre 1 cm y 4 cm tienen el mismo ancho de pulso de eco. A partir de 4 cm el pulso comenzaba a aumentar proporcionalmente para cada incremento de distancia a medir.

Por lo tanto se deduce que la distancia mínima no es la mostrada teóricamente por el fabricante, sino que son 4 cm.

La solución es cambiar en el código fuente el valor de distancia mínima, modificando el valor de la etiqueta "MINIMA_DISTANCIA" a 4 (MINIMA_DISTANCIA EQU .4)

4.2.2 Ajuste del Timer 0

Una mejora, y la más importante, es el ajuste exacto del Timer 0 para así optimizar la medida con el menor error posible. Para ello tomamos la siguiente relación de medidas:

Distancia a Medir (cm)	Tiempo de pulso de eco (µs)	Incremento de tiempo de pulso de eco respecto a la anterior medida (µs)
4	271	N/A
5	327	56
6	380	53
7	438	58
8	497	59
9	551	54
10	605	54

Tabla 4.1 *Medidas reales para optimizar la medida*

Haciendo una media de estas distancias, se puede comprobar que por cada centímetro medido, el pulso de eco se incrementa uno 55 µs (aprox.). Por lo tanto no cumple exactamente con las especificaciones del fabricante, ya que nuestro cálculo inicial es para 60 µs. Por lo tanto utilizando la fórmula (3.3) calcularemos el valor de "CargaTMR0" para un incremento de 55 µs.

No obstante, para una mejor verificación de este tiempo, se utilizó el siguiente programa:

```
;******* Timer0_pruebas.asm ********
; El cálculo de la carga del TMR0 se hará de forma que se tenga en cuenta los tiempos
; de las instrucciones para conseguir tiempos exactos. Para calcular los valores de carga
; del TMR0 hay que ayudarse del simulador del MPLAB y de la ventana de reloj
Stopwatch.
;
; ZONA DE DATOS ************

LIST            P=16F84A
INCLUDE                 <P16F84A.INC>
__CONFIG        _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC

CBLOCK          0x0C
ENDC

#DEFINE          Salida   PORTB,3

; ZONA DE CÓDIGOS *********

ORG     0
Inicio
        bsf     STATUS,RP0      ; Acceso al Banco 1.
        bcf     Salida          ; Esta línea se configura como salida.
        movlw   b'00000000'
        movwf   OPTION_REG      ; Prescaler de 2 para el TMR0.
        bcf     STATUS,RP0      ; Acceso al Banco 0.
Principal
        bsf     Salida          ; La salida pasa a nivel alto
        call    Timer0_55us     ; durante este tiempo.
        bcf     Salida          ; La salida pasa a nivel bajo
        call    Timer0_55us     ; durante este tiempo.
        goto    Principal
;
; Subrutina "Timer0_55us" -----
;
; Con el simulador se comprueba que se obtienen un tiempo bastante aproximado de
; 55us.
;
```

```
TMR0_Carga55us      EQU    d'231'  ; Este valor es el resultado de la fórmula 3.3
Timer0_55us
        movlw  TMR0_Carga55us      ; Carga el Timer 0.
        movwf  TMR0
        bcf    INTCON,T0IF         ; Resetea el flag de desbordamiento del TMR0.
        Timer0_Desbordamiento
        btfss  INTCON,T0IF         ; ¿Se ha producido desbordamiento?
        goto   Timer0_Desbordamiento       ; Todavía no. Repite.
        return

END
```

Gracias a este pequeño programa (totalmente independiente del programa principal que controla el medidor de distancias por ultrasonidos), y a las herramientas proporcionadas por el MPLAB como, el simulador de código, los puntos de ruptura (*Breakpoints*) y a la ventana "Stopwatch", podemos verificar que el valor calculado de "231" provoca un desbordamiento en el Timer 0 aproximadamente cada 55µs. A continuación se demuestra en las siguientes figuras:

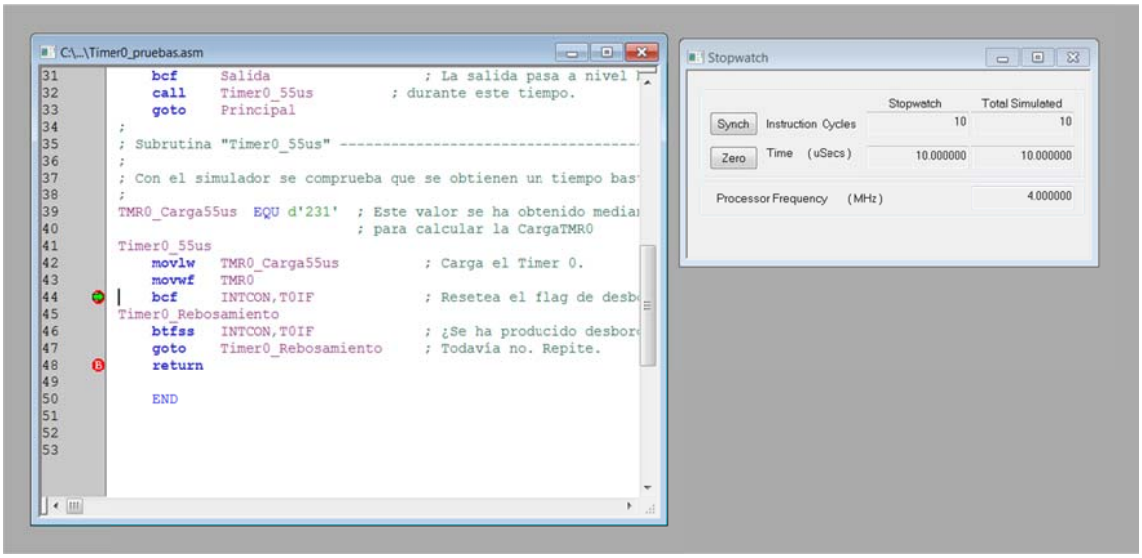


Figura 4.5 *Verificación de los 55µs antes de entrar en el bucle "Timer0_Desbordamiento"*

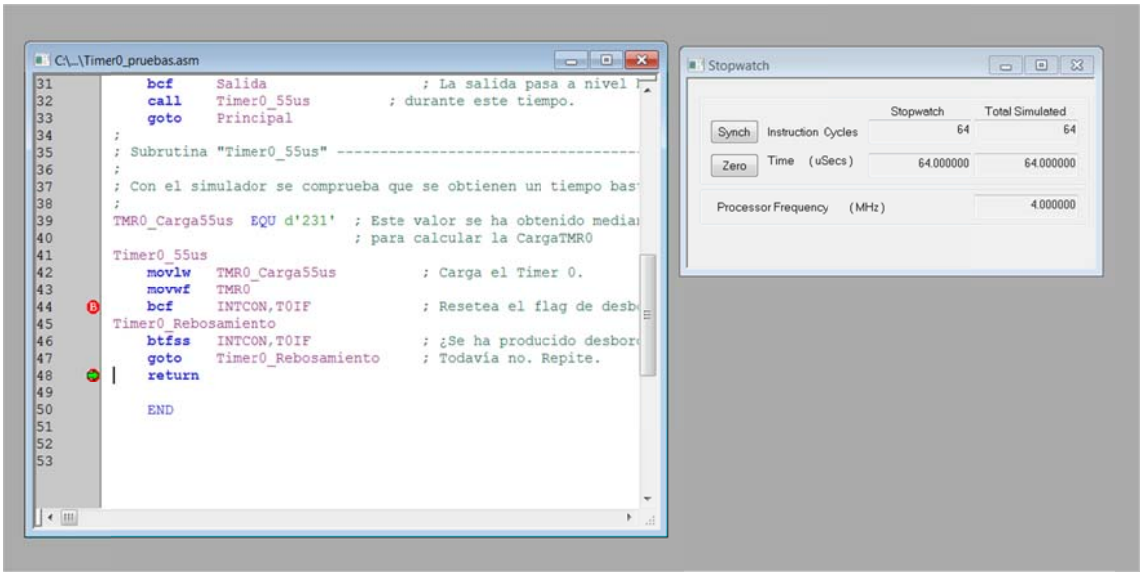


Figura 4.6 *Verificación de los 55µs una vez finalizado el bucle "Timer0_Desbordamiento"*

En las figuras 4.5 y 4.6 se puede ver perfectamente como el tiempo antes de entrar en el bucle es de 10µs y cuando acaba es de 64µs, por lo tanto la diferencia es de 54µs. Esto nos garantiza que nuestro valor cargado en la variable "TMR0_Carga55us" es prácticamente fiable y correcto.

4.2.3 Verificación del Sensor

Esta mejora consiste en mostrar un mensaje por pantalla cuando el sensor no emite el flanco de subida del pulso de eco. Si no se hubiera realizado esta mejora, en el módulo LCD quedaría permanentemente mostrando el mensaje de "Midiendo…" y programa permanecería 'colgado' en ese punto sin continuar ejecutando código.

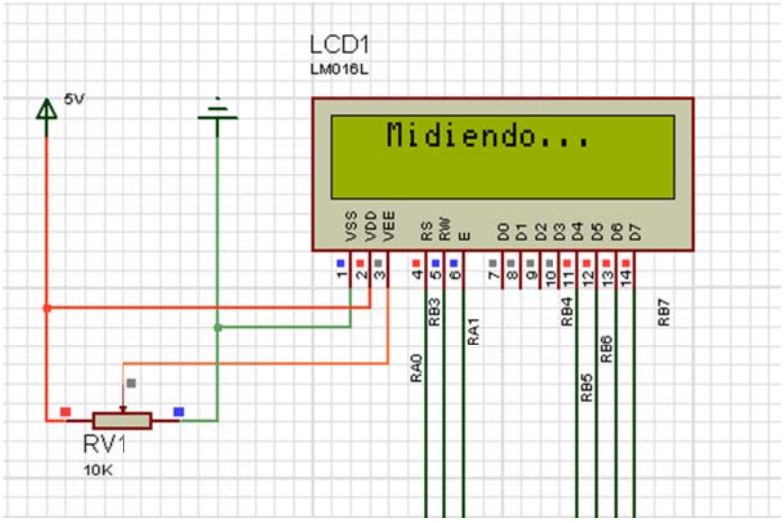


Figura 4.7 *Estado permanente del programa si el sensor no funciona*

Para realizar esta mejora se añadió las siguientes líneas de código al programa:

Código añadido a la subrutina de "Mensajes":

```
MensajeErrorSensor
        DT "    Error",            0x00
MensajeErrorSensor2
        DT " en el Sensor",        0x00
MensajeComprobarFuncionamiento
        DT "  Verificar",          0x00
MensajeComprobarFuncionamiento2
        DT "Funcionamiento",   0x00
```

Código añadido al programa:

```
Comprobacion_Funcionamiento_Sensor
        decfsz  ContadorVerificarSensor,1
        goto    Espera_Inicio_Eco
        goto    Error_Funcionamiento_Sensor

Error_Funcionamiento_Sensor
        call    LCD_Borra
        movlw   MensajeErrorSensor
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeErrorSensor2
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MensajeComprobarFuncionamiento
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeComprobarFuncionamiento2
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        call    Retardo_2s
        goto    Principal
```

- En el Anexo se puede encontrar la estructura completa del programa final y una explicación detallada de cada línea de código.

Esta parte del código se encarga de comprobar que se ha generado correctamente el flanco de subida del pulso de eco. Para ello se creó la variable "ContadorVerificarSensor" iniciándola con un valor de 250. En la figura 4.4 podemos observar que entre el final del pulso de disparo y el inicio del pulso de eco hay un tiempo de 200µs (aprox.). Por lo tanto estas líneas de código comprobaran si se ha generado el pulso de eco antes de 250µs (de ahí el valor 250 cargado en la variable). Si el pulso no se ha iniciado antes de ese tiempo automáticamente el programa saltará a la subrutina "Error_Funcionamiento_Sensor", mostrando en el módulo LCD los siguientes mensajes:

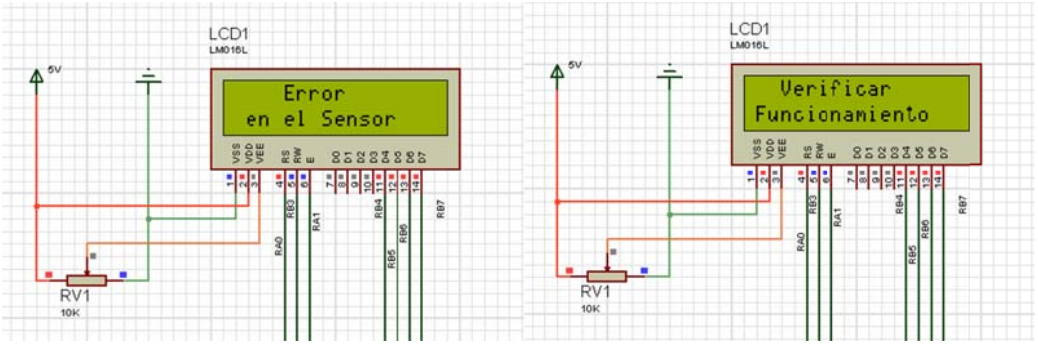


Figura 4.8 *Mensajes de errores mostrados en el LCD*

Capítulo 5. Diseño del Montaje Final

5.1 Diseño del Layout

Una vez realizadas todas las mejoras y viendo que todo el funcionamiento es correcto, se prosiguió a realizar el PCB (*Printed Circuit Board*) de la figura 4.2.

Para la realización del PCB también se utilizó el software de desarrollo PROTEUS, esta vez su vertiente ARES, que es la que permite la posibilidad de diseñar *layouts*.

En las siguientes figuras se muestra el desarrollo del circuito PCB del medidor de distancias por ultrasonidos:

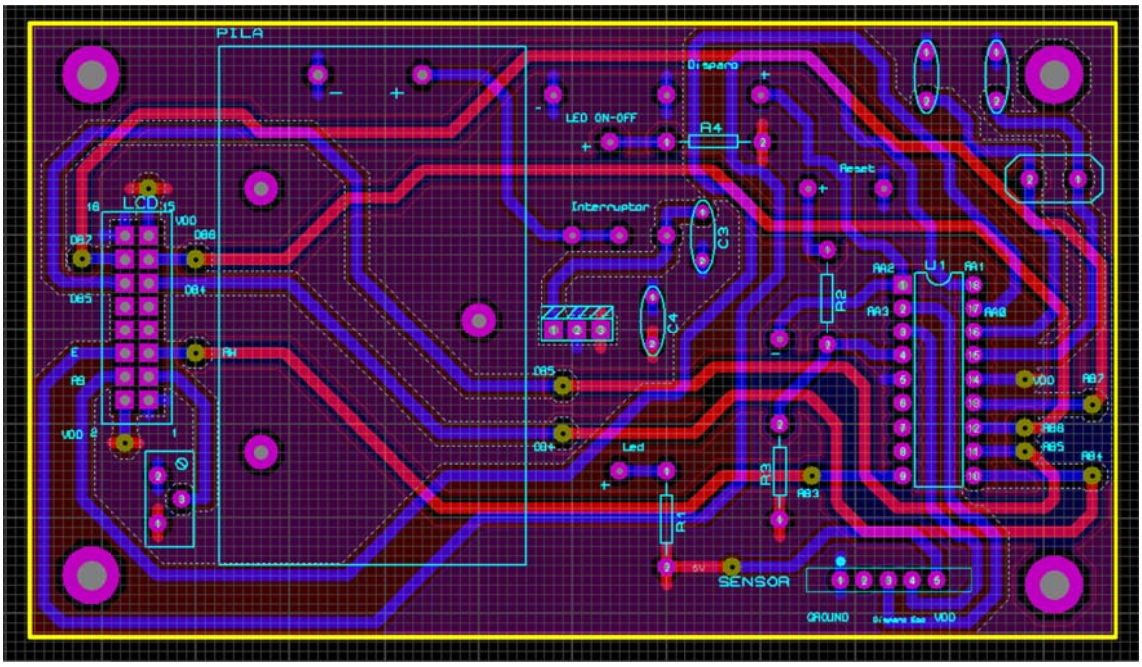


Figura 5.1 *Vista del PCB en PROTEUS*

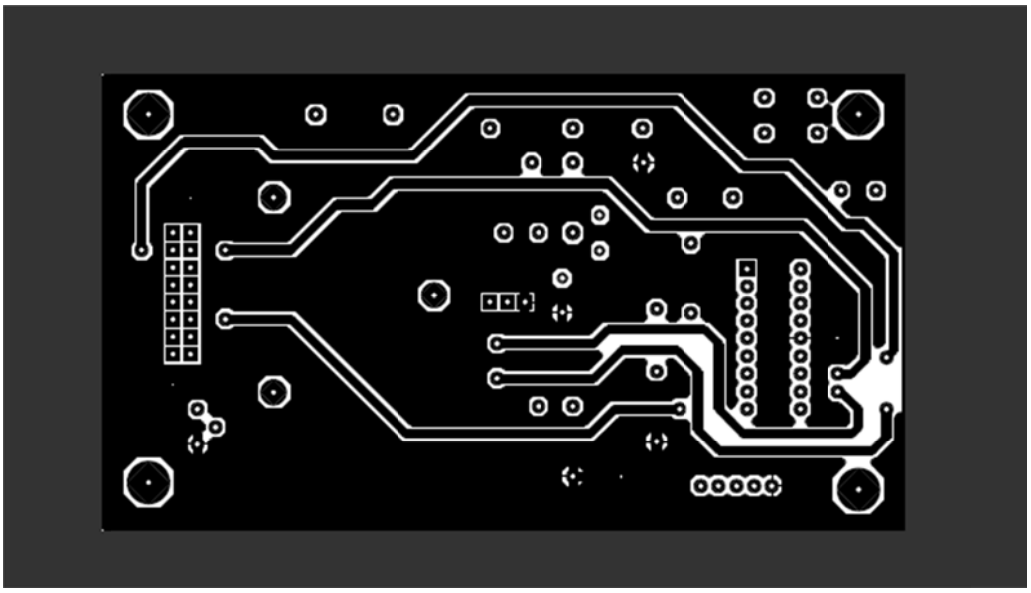


Figura 5.2 *Visualización de las pistas de la capa superior (Top Cooper)*

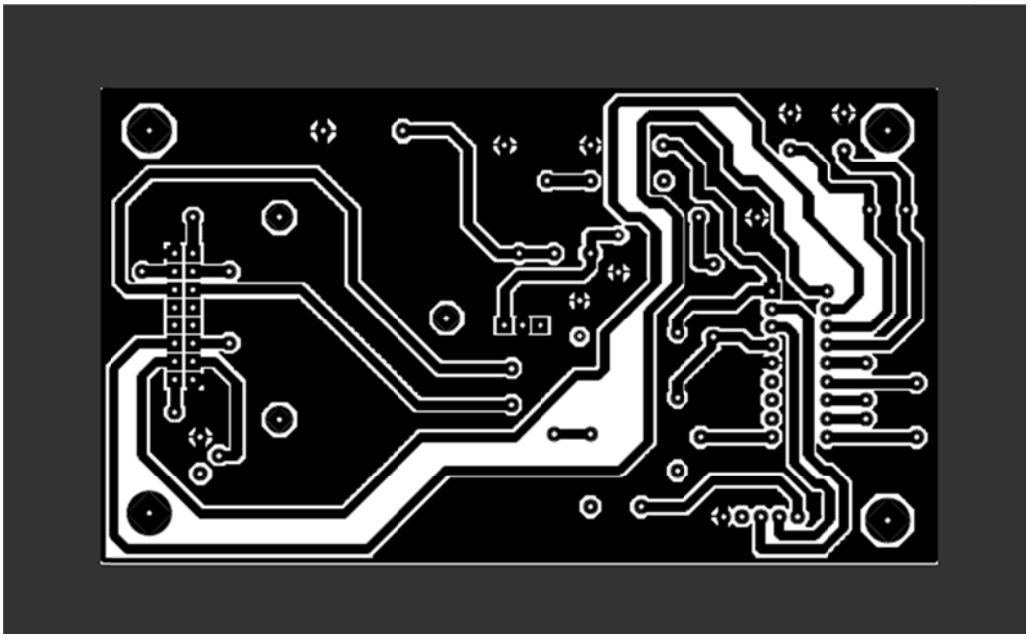


Figura 5.3 *Visualización de las pistas de la capa inferior (Bottom Cooper)*

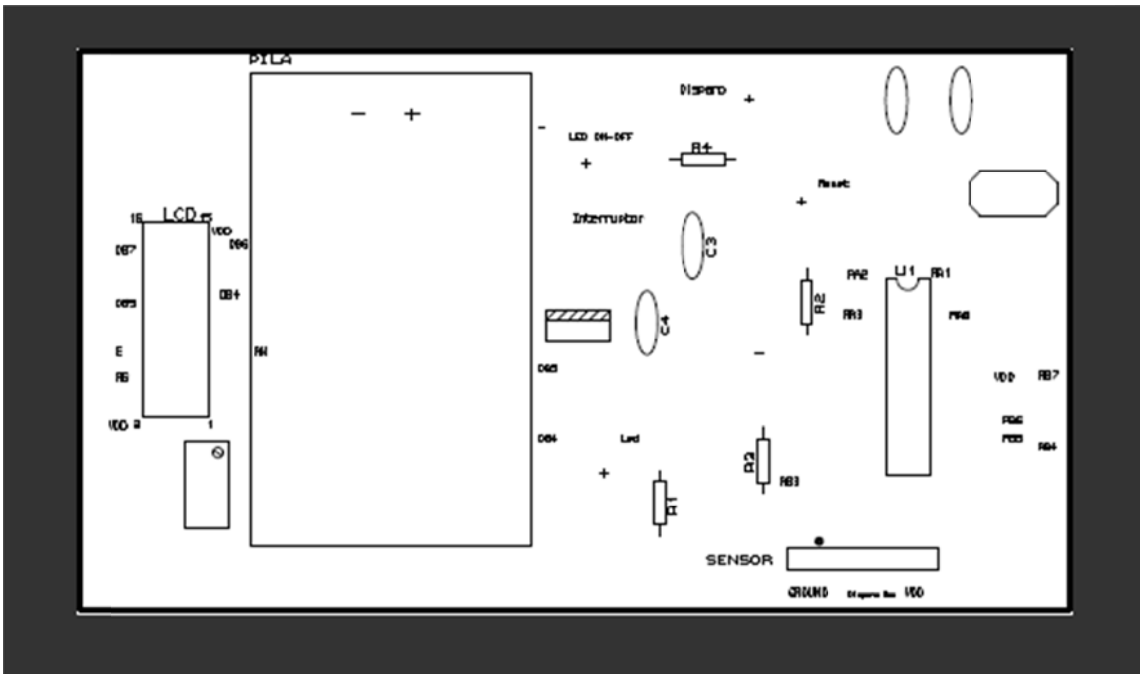


Figura 5.4 *Visualización de la situación de los componentes en la placa*

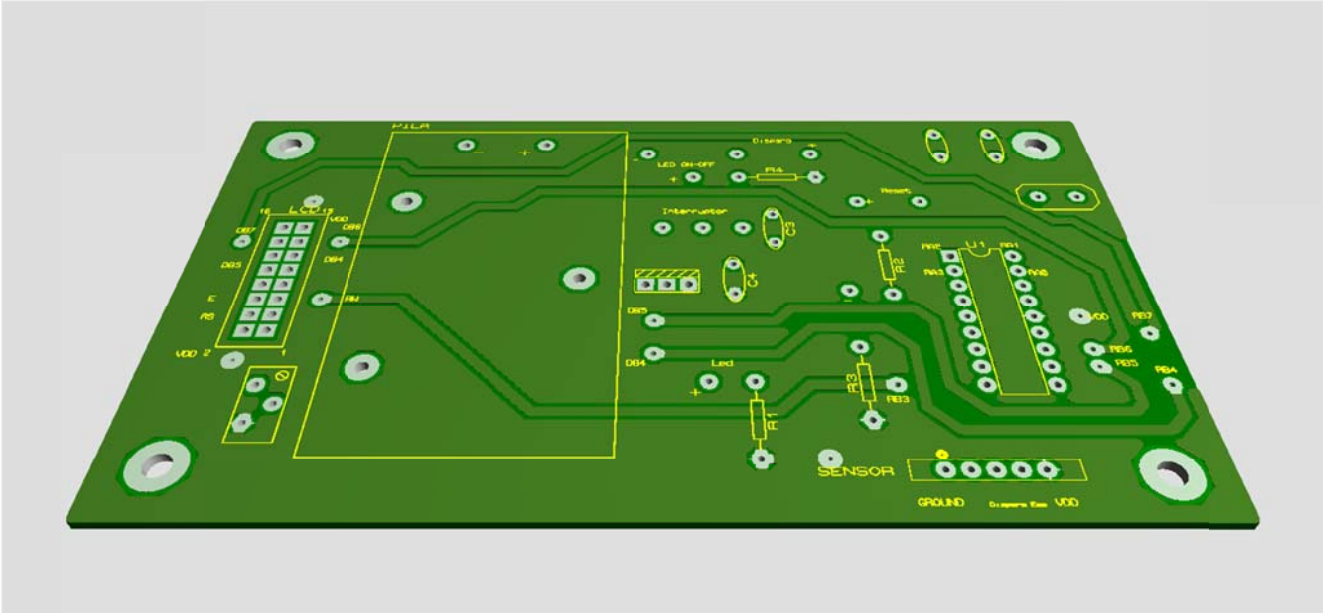


Figura 5.5 *Visualización en 3D de la placa (Top Cooper)*

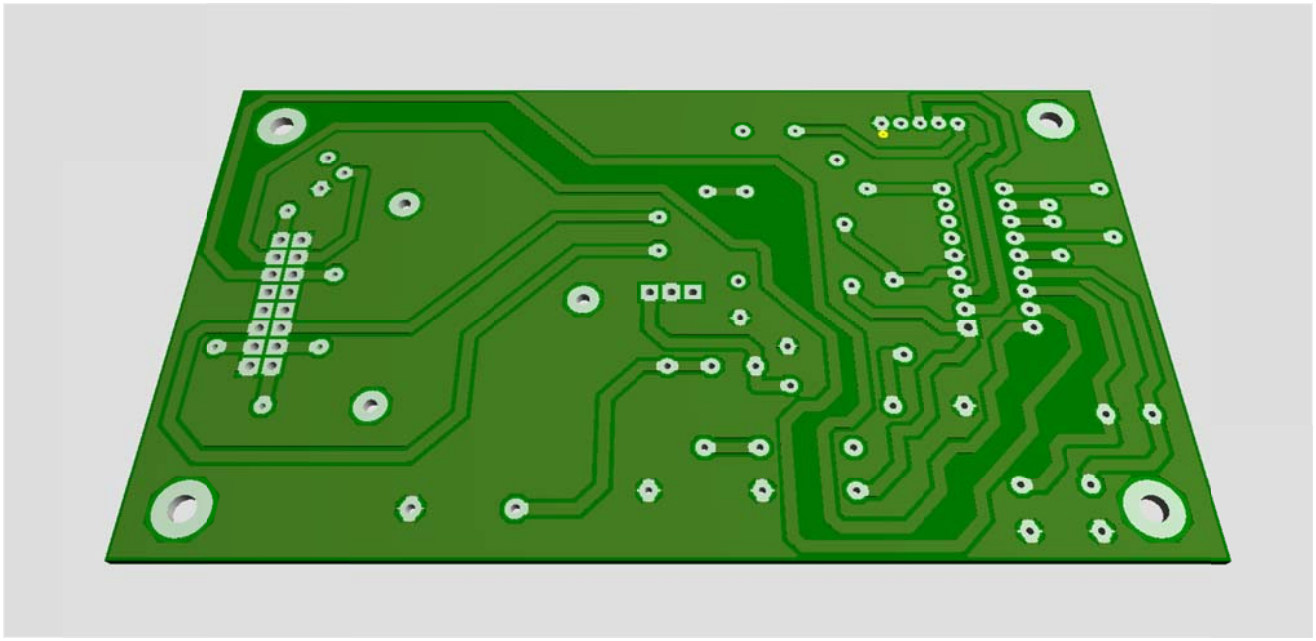


Figura 5.6 *Visualización en 3D de la placa (Bottom Cooper)*

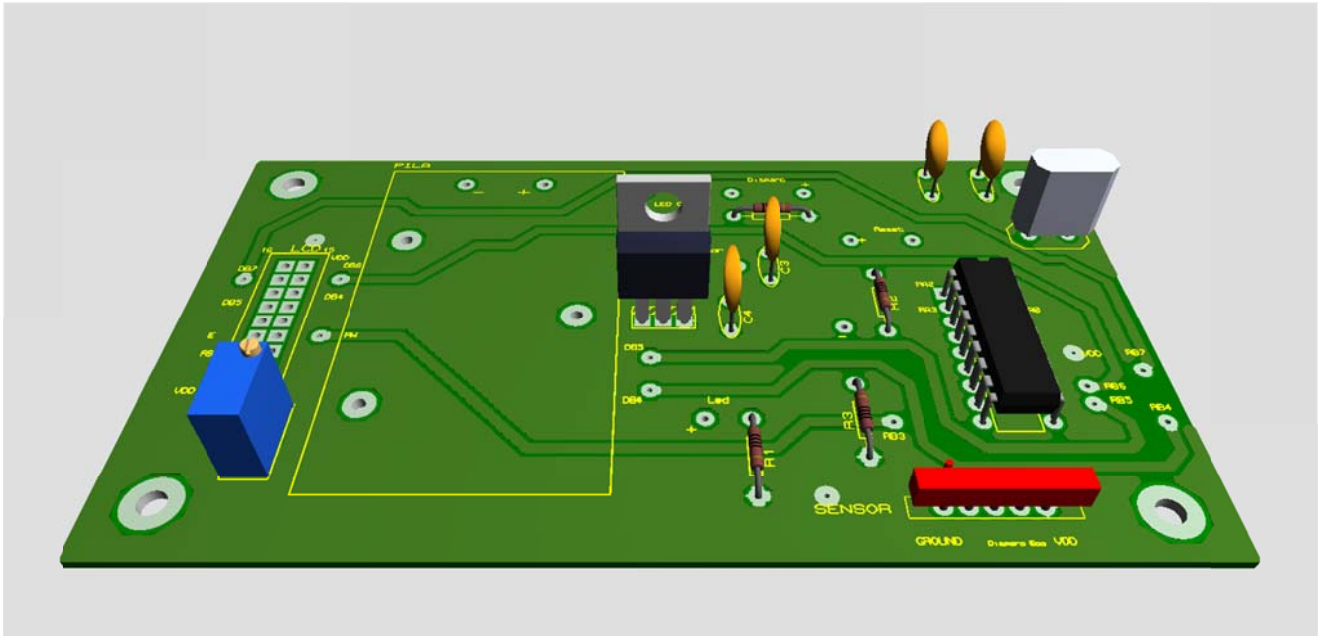


Figura 5.7 *Visualización en 3D de la placa con los componentes (Top Cooper)*

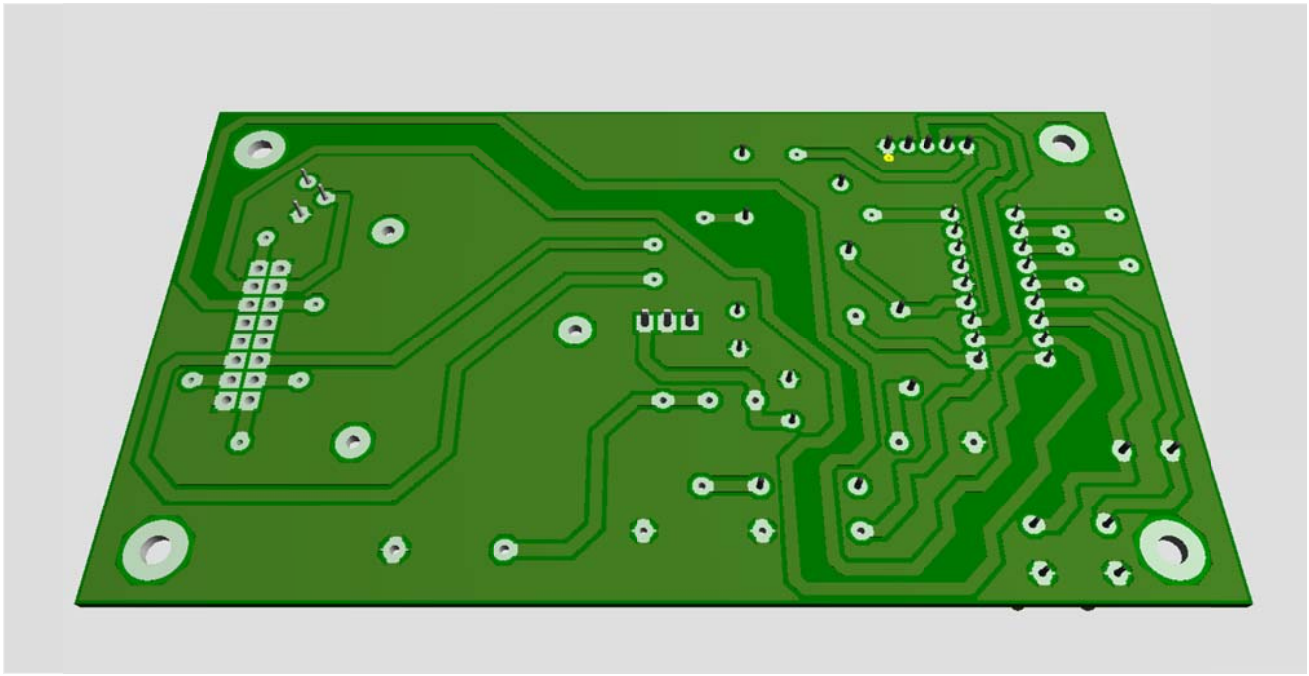


Figura 5.8 *Visualización en 3D de la placa con los componentes (Bottom Cooper)*

5.2 Estructura Final

A continuación se muestran una serie de imágenes del proceso de construcción del proyecto:

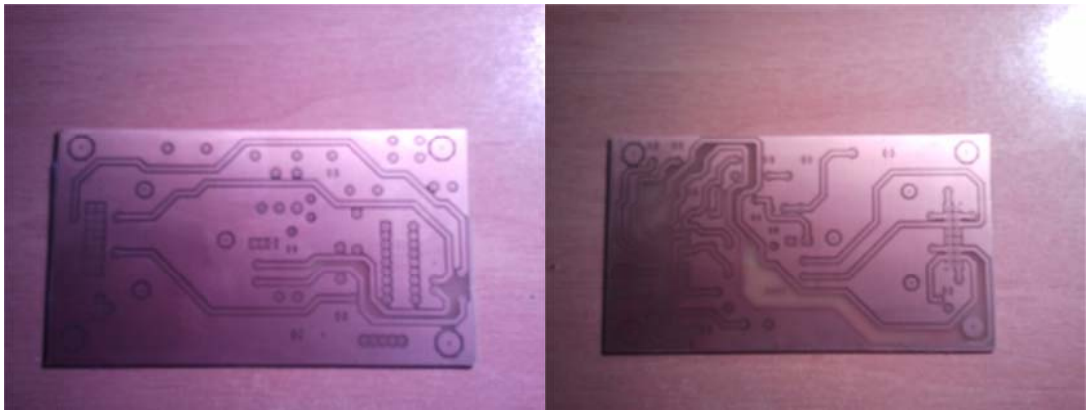


Figura 5.9 *Vistas de la placa de circuito impreso. En la izquierda las pistas correspondientes a la Top Copper y a la derecha las pistas correspondientes a Bottom Copper.*

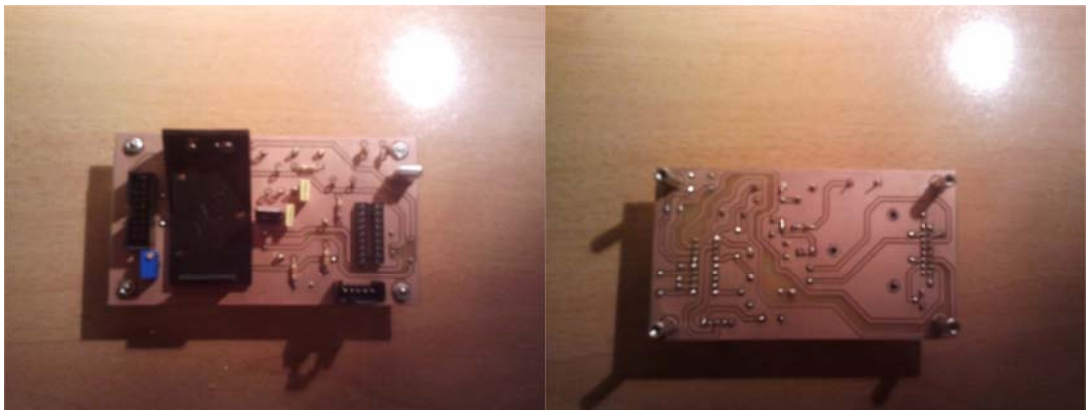


Figura 5.10 *A la izquierda la placa de circuito impreso con los componentes. A la derecha la placa con las soldaduras realizadas.*

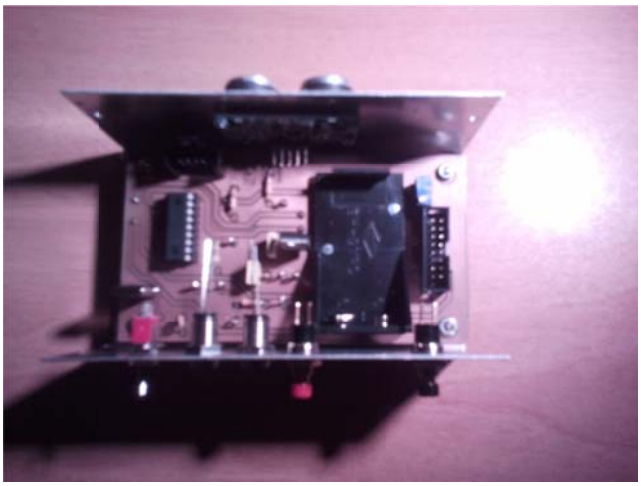


Figura 5.11 *Placa de circuito impreso ubicada dentro de la caja de metal junto con los pulsadores, los led's, el interruptor y el sensor*

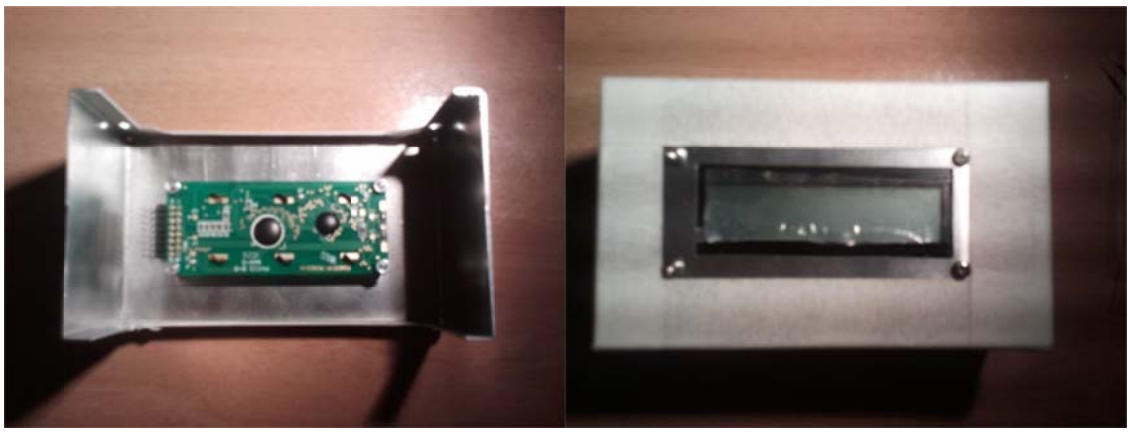


Figura 5.12 *Vistas del módulo LCD insertado en la caja de metal*

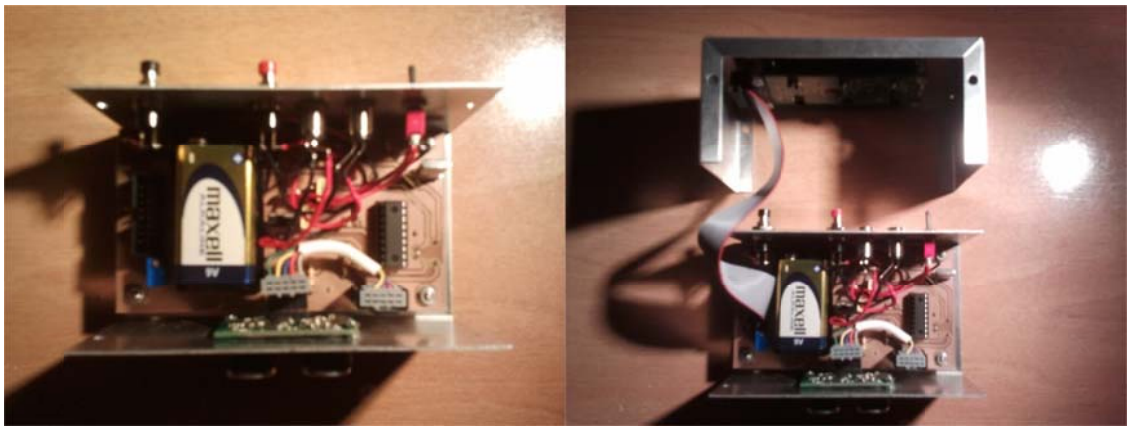


Figura 5.13 *Vista del conexionado entres los componentes de la placa y el resto de dispositivos*

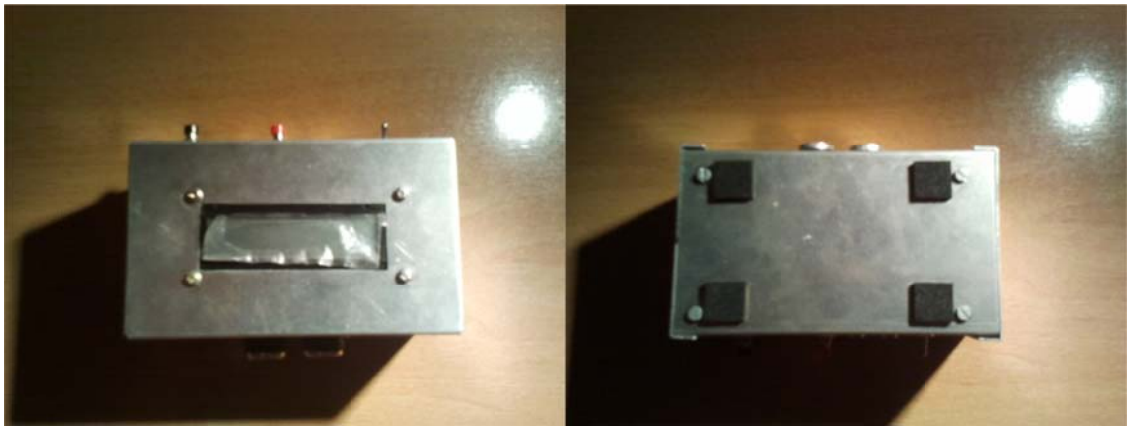


Figura 5.14 *Vista superior e inferior del medidor de distancias por ultrasonidos*

Figura 5.15 *Vista frontal y posterior del medidor de distancias por ultrasonidos*

Capítulo 6. Características, Medidas y Presupuesto del Diseño

6.1 Características del Medidor de Distancias por Ultrasonidos

A continuación se muestra una tabla con las características básicas del prototipo diseñado:

Sistema de Alimentación	Pila 9V
Consumo del Dispositivo	261 mW
Frecuencia de los Impulsos Ultrasónicos	40 KHz
Distancia Mínima	4 cm
Distancia Máxima	250 cm
Resolución	1 cm
Margen de Error	±1 cm
Dimensiones	125 mm x 60 mm x 75 mm
Ancho del Haz del Sensor	π/4 (45º)
Cobertura Transversal Máxima*	1,96 m $\left(\frac{\pi}{4} \cdot 250\ cm\right)$

Tabla 6.1 *Características del Medidor de Distancias por Ultrasonidos*

*El dispositivo siempre detectará el objeto u obstáculo más cercano al sensor dentro de un arco de circunferencia de 1,96 m respecto al centro del sensor. Pudiendo dar medidas no deseadas.

6.2 Tabla de Medidas

A continuación se muestra una tabla con diez medidas realizadas y el cálculo del error relativo de cada una de ellas, donde el valor relativo es:

$$\boldsymbol{Error\ Relativo}\ (\%) = \left[\frac{\boldsymbol{Dist.Medida - Dist.Real}}{\boldsymbol{Dist.Real}}\right] \cdot \mathbf{100} \qquad \textbf{(6.1)}$$

Distancia Real (cm)	Distancia Medida (cm)	Error Relativo (%)
25	25	0
50	51	2
75	76	1,33
100	100	0
125	124	0,8
150	150	0
175	174	0,57
200	200	0
225	224	0,44
250	249	0,4

Tabla 6.2 *Medidas realizadas*

6.3 Presupuesto del Prototipo

A continuación se detalla una lista con el precio de los materiales y dispositivos utilizados:

Componente o Dispositivo	Descripción	Precio por Unidad (€)	Cantidad	Total (€)
SRF04	Sensor de Ultrasonidos	21	1	21
PIC16F84A	Microcontrolador de 8 bits	4,47	1	4,47
PC1602-H	Display Alfanumérico de 2 líneas por 16 caracteres por línea	20,40	1	20,40
7805	Regulador de Tensión de 9V a 5V	1,28	1	1,28
Placa Circuito Impreso	Placa de Fibra de Vidrio con dimensiones 115mm x 68mm	3,43	1	3,43
Resistencias	¼ W y 5% de tolerancia. Con valores de 100Ω, 330Ω y 10K	0,012	4 (dos de 330, una de 10 y otra de 100)	0,048
Potenciómetro Multivuelta	Resistencia variable de 10K	0,81	1	0,81
LED	Diodo emisor de luz de color rojo y de color verde, para poder ser incorporado en caja	1,42	2	2,84
Pulsador	Pulsador de color negro y de color rojo, para poder ser incorporado en caja	2,10	2	4,2
Interruptor	Interruptor de dos posiciones con una paso de 2,54mm entre pin y pin	1,66	1	1,66
Pila	9V	3,35	1	3,35
Soporte Pila	Soporte de plástico para ubicar la pila de 9V en la placa	1,08	1	1,08
Caja	Caja de aluminio con dimensiones 125 mm x 60 mm x 75 mm	6,48	1	6,48
Separadores	Separadores hexagonales para separar la placa de la caja	0,12	4	0,48
Condensador Cerámico	Valor de 22pF	0,12	2	0,24
Condensador Poliester	Valor de 100nF y 330nF	0,20	2 (uno de cada valor)	0,40
Cristal	Cristal de cuarzo de 4 MHz	1,02	1	1,02
Tira poste	Tira pin macho torneado	1,45	1	1,45
Tira poste	Tira pin hembra torneado	1,08	1	1,08
Tira poste	Tira pin macho doble	4,48	1	4,48

Tira poste	Tira pin hembra acodado	2,80	1	2,80
Tira poste	Tira pin hembra doble acodado	4,48	1	4,48
Zócalo	Zócalo torneado de 18 pines	1,02	1	1,02
Terminales	Terminales planos para cables entre placa y componentes	0,10	11	1,1
Conector LCD	Conector macho recto doble de 16 pines para LCD	0,69	2	1,38
Conector LCD	Conector hembra cable plano doble de 16 pines para LCD	0,44	2	0,88
Conector Sensor	Conector macho de 5 pines para comunicación con el sensor	0,53	1	0,53
Conector Sensor	Conector hembra de 5 pines para comunicación con el sensor	0,65	2	1,3
Juego Tornillo/Tuerca	Tornillo y tuerca de 5 mm de diámetro para unir la placa con los separadores	0,049	10	0,49
Tornillo Rosca-Chapa	Tornillo de 5 mm para cerrar la caja	0,075	4	0,3
Cable Conexión	Cable estañado para conexionado entre componentes y los terminales	0,102	11	1,12
Cable Sensor	Cable grueso con 5 hilos para comunicación entre sensor y placa	0,71	1 metro	0,71
Cable LCD	Cable plano de 16 hilos para conexionado entre el LCD y la placa	0,80	1 metro	0,80
Macarrón Termorretráctil	Macarrón rojo y macarrón negro de 2 mm de diámetro	0,93	2 (1 metro de cada color)	1,86
Separadores Espuma	Separadores de espuma para el aislamiento de la caja	0,145	4	0,58
				99,54*

Tabla 6.3 *Presupuesto detallado del Medidor de Distancias por Ultrasonidos*

* Este precio es aproximado debido a que se realiza por cálculo de cada componente. A la hora de realizar la compra el precio puede aumentar ya que algunos componentes no se pueden comprar por unidades, sino que tienen que ser comprado en paquetes de varias unidades.

Conclusiones

La conclusión principal es que se han conseguido cumplir todos los objetivos uno por uno. Pudiendo realizar el diseño de un dispositivo desde los inicios hasta el final. Estudiando el comportamiento de cada elemento, aprendiendo el lenguaje de programación utilizado, utilizando nuevos software de desarrollo, probar el dispositivo y construirlo para poder ser presentado. Todo ello invirtiendo muchas horas de trabajo y dedicación para obtener finalmente un dispositivo totalmente fiable y práctico, utilizando los mínimos materiales necesarios para su realización.

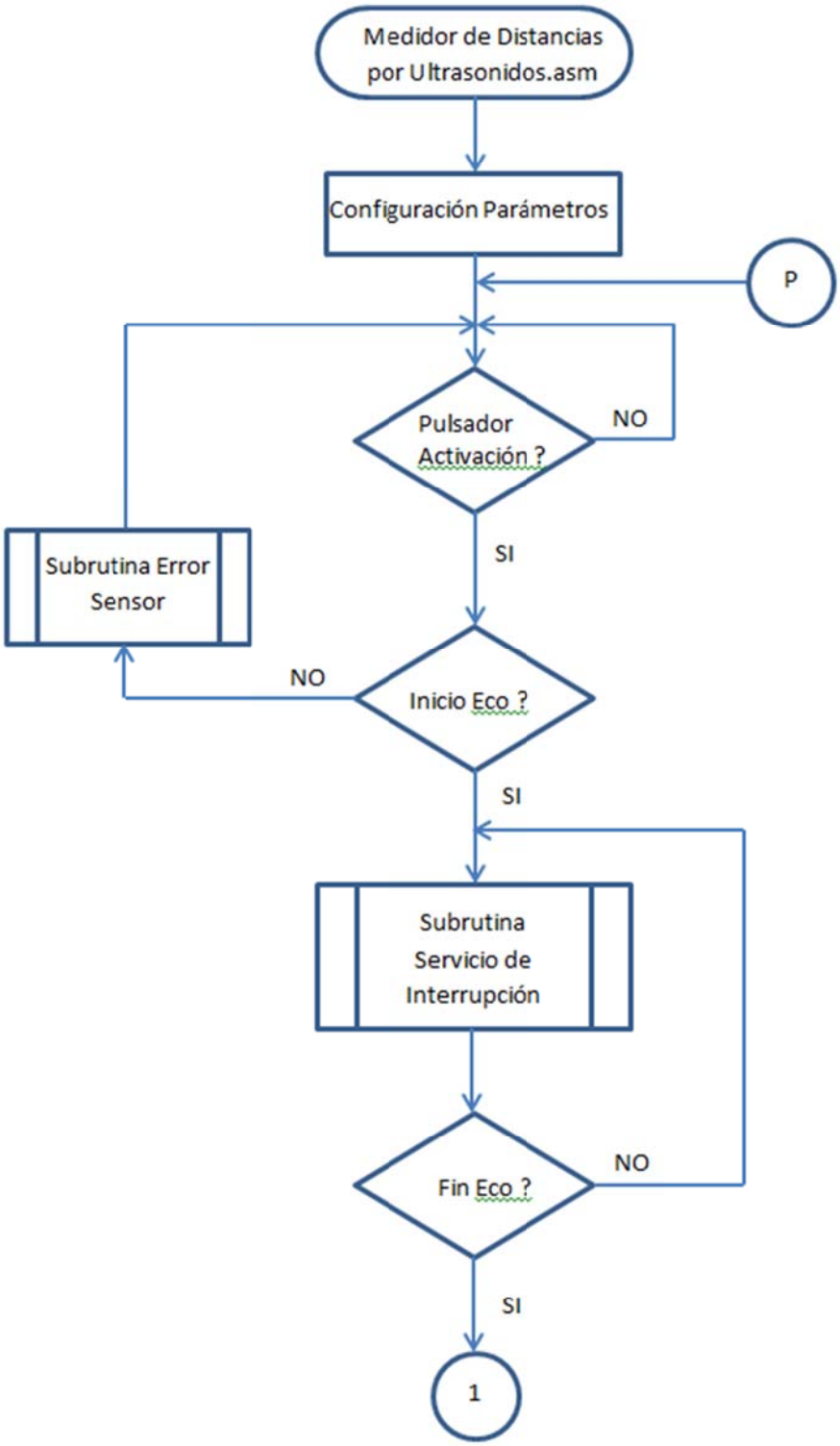
El principal inconveniente de este proyecto es que debido al ancho de haz del sensor, que es de unos 45^{o}, las aplicaciones para las cuales se podría utilizar este diseño están limitadas en cuanto a resolución transversal. Ya que un obstáculo que se encontrará dentro del campo de medida y que estuviera más cerca del blanco, nos daría como medida la del obstáculo en lugar de la del blanco ya que el eco que proviene del obstáculo llegaría antes al sensor, provocando una medida no deseada. Una forma de evitar este inconveniente sería utilizar otro sensor de ultrasonidos con un ancho de haz más estrecho, como por el ejemplo el sensor SRF235, pero incrementando considerablemente el precio de coste del diseño. Otra alternativa a esto, para aplicaciones de medir distancias largas, sería utilizar sensores láser o de radiofrecuencia.

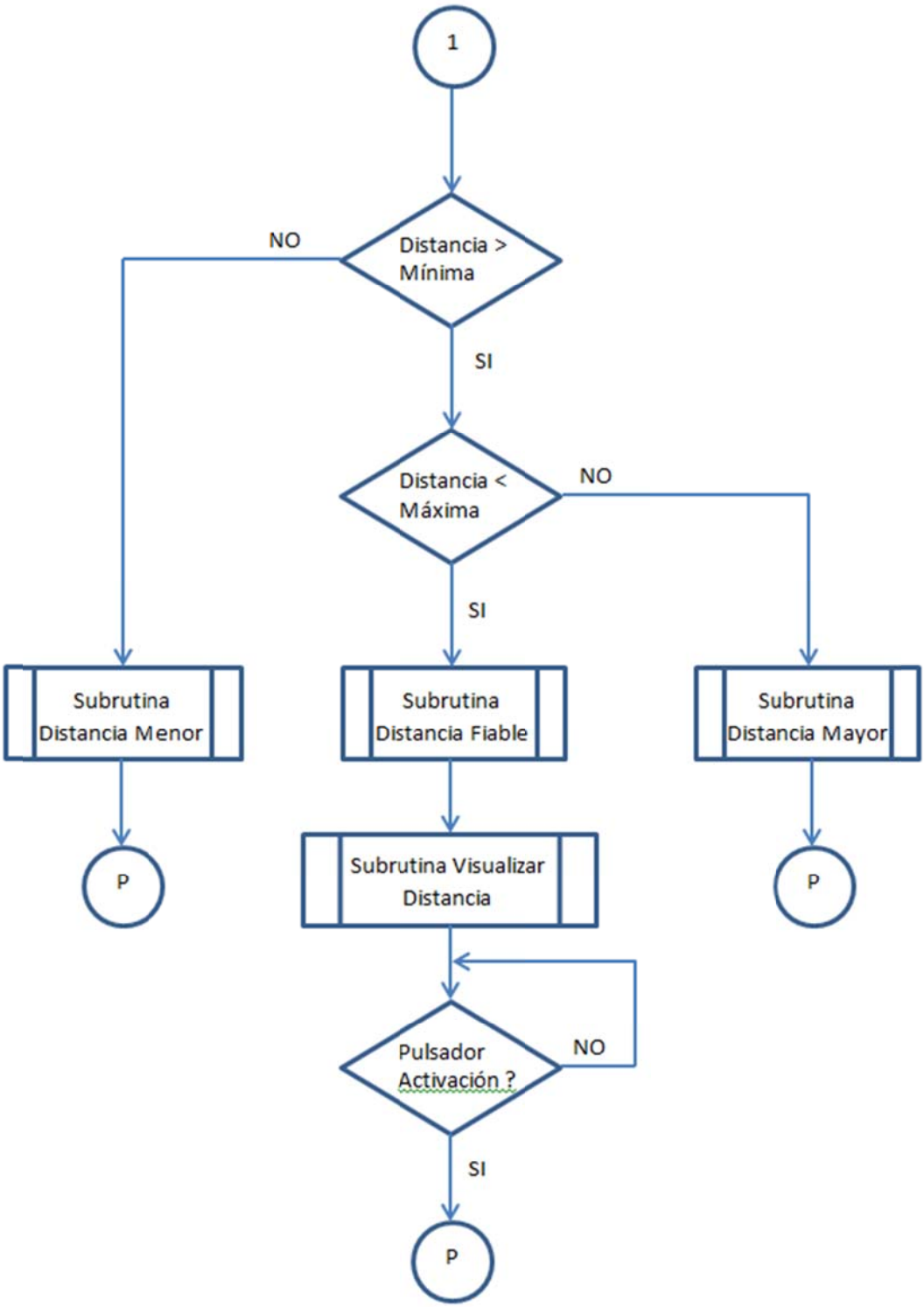
Todo este trabajo me ha servido básicamente para aprender, para adquirir nuevos conocimientos y recuperar los que ya tenía olvidados.

ANEXO I

(Código Fuente)

Flujograma


Medidor de Distancias por Ultrasonidos.asm
Configuración Parámetros
P
Pulsador Activación ?
NO
SI
Subrutina Error Sensor
NO
Inicio Eco ?
SI
Subrutina Servicio de Interrupción
Fin Eco ?
NO
SI
1


1
Distancia > Mínima
NO
SI
Distancia < Máxima
NO
SI
Subrutina Distancia Menor
Subrutina Distancia Fiable
Subrutina Distancia Mayor
P
Subrutina Visualizar Distancia
P
Pulsador Activación ?
NO
SI
P

Programa Principal

```
;***************Medidor de distancias por ultrasonidos.asm *************************
;
; Programa para un medidor de distancias hasta un objeto utilizando sensor por ultrasonido
; SRF04.
;
; ZONA DE DATOS ******************************************************************

        LIST            P=16F84A
        INCLUDE         <P16F84A.INC>
        __CONFIG        _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC  ;Parámetros de configuración.

        CBLOCK  0x0C
        Distancia    ; Registro donde quedará guardada la distancia, dicha medida se expresará en centímetros.
        ContadorVerificarSensor    ; Registro donde se comprobará que el funcionamiento del sensor es correcto.
        ENDC

#DEFINE  PulsadorActivacion PORTA,2          ; Activación para activar la señal de disparo, en la entrada RA2.
#DEFINE  Disparo            PORTA,3          ; Disparo para iniciar la medida, en la entrada RA3.
#DEFINE  Eco                PORTA,4          ; Pulso cuya anchura hay que medir, en la entrada RA4.

MINIMA_DISTANCIA    EQU    .4      ; Se le asigna a esta etiqueta el valor en decimal de la distancia mínima que
                                   ; puede medir el sensor (4 cm).
MAXIMA_DISTANCIA  EQU    .250     ; Se le asigna a esta etiqueta el valor en decimal de la distancia
                                   ; máxima que puede medir el sensor (250 cm).
TMR0_Carga55micros EQU   .231     ; Se le asigna a esta etiqueta el valor calculado teóricamente a partir de los
                                   ; datos del fabricante y ajustado mediante mediciones para crear una
                                   ; temporización y así provocar una interrupción en el Timer 0 cada 55us.
NUMERO_VECES_VERIFICAR_SENSOR  EQU .250   ; Se le asigna a esta etiqueta el valor para verificar el
                                          ; funcionamiento del sensor al circuito mediante control de
                                          ; tiempo. Dicho control ha sido verificado mediante osciloscopio
                                          ; calculando el tiempo entre el final del pulso de Disparo y
                                          ; el inicio del pulso de Eco.

; ZONA DE CÓDIGOS ****************************************************************

        ORG     0                         ; Indica la dirección de inicio del programa.
        goto    Inicio
        ORG     4                         ; Indica la dirección de comienzo de la interrupción.
        goto    Interrupcion_Timer0_55us

Mensajes
        addwf   PCL,F                     ; Contador indexado.
MensajeDistancia
        DT "Dist.:  ",              0x00  ; Tabla para mostrar "Dist.:".
MensajeDistancia2
        DT "   Distancia",          0x00  ; Tabla para mostrar la palabra "Distancia".
MensajeCentimetro
        DT " cm",                   0x00  ; Tabla para mostrar "cm".
MensajeDistanciaMenor
        DT "Dist. Menor de:",       0x00  ; Tabla para mostrar "Dist. Menor de".
MensajeDistanciaMenor2
        DT "    4 cm",              0x00  ; Tabla para mostrar "4 cm".
MensajeDistanciaMayor
        DT "Dist. Mayor de:",       0x00  ; Tabla para mostrar la palabra "Dist. Mayor de".
MensajeDistanciaMayor2
        DT "    250 cm",            0x00  ; Tabla para mostrar "250 cm".
```

```
MensajeFueraRango
        DT "Fuera del Rango",       0x00    ; Tabla para mostrar "Fuera de Rango".
MensajeDentroRango
        DT "Dentro del Rango",      0x00    ; Tabla para mostrar "Dentro del Rango".
MensajeInicioMedida
        DT "  Para Medir",          0x00    ; Tabla para mostrar "Iniciar Medida".
MensajeMidiendo
        DT "  Midiendo...",         0x00    ; Tabla para mostrar "Midiendo...";
MensajeDisparo
        DT " Pulsar Disparo",       0x00    ; Tabla para mostrar "Pulsar Disparo";
MensajeErrorSensor
        DT "    Error",             0x00    ; Tabla para mostrar "Error";
MensajeErrorSensor2
        DT " en el Sensor",         0x00    ; Tabla para mostrar "en el Sensor";
MensajeComprobarFuncionamiento
        DT "  Verificar",           0x00    ; Tabla para mostrar "Verificar";
MensajeComprobarFuncionamiento2
        DT "Funcionamiento",        0x00    ; Tabla para mostrar "Funcionamiento";
MensajePulsarParaContinuar
        DT "Pulsar Disparo",        0x00    ; Tabla para mostrar "Pulsar Disparo";
Inicio
        call    LCD_Inicializa
        bsf     STATUS,RP0          ; Vamos al Banco 1.
        bsf     PulsadorActivacion  ; Configuramos el pin RA2 como entrada.
        bcf     Disparo             ; Configuramos el pin RA3 como salida.
        bsf     Eco                 ; Configuramos el pin RA4 como entrada.
        movlw   b'00000000'         ; Configuramos el registro OPTION_REG con un divisor de frecuencia por 2
                                    ; para el TMR0, el flag T0SE configurado a 0 para ser incrementado el TMR0
                                    ; cada flanco de subida y el flag T0CS a 0 para configurar el TMR0 como
                                    ; temporizador.
        movwf   OPTION_REG
        bcf     STATUS,RP0          ; Vamos al Banco 0.
        bcf     Disparo             ; Inicializa línea de Disparo en bajo.
Principal
        clrf    Distancia                   ; Inicializa el registro a 0.
        clrf    ContadorVerificarSensor     ; Inicializa el registro a 0.
        movlw   NUMERO_VECES_VERIFICAR_SENSOR           ; Cargamos el registro de
                                                        ; ContadorVerificarSensor con un valor de 250.
        movwf   ContadorVerificarSensor
        movlw   MensajeInicioMedida
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeDisparo
        call    LCD_Mensaje
Activacion_Disparo
        btfsc   PulsadorActivacion  ; Hasta que no hay un 0 en la entrada de activación, el micro no activará
        goto    Activacion_Disparo  ; el sensor mediante la señal de disparo.
        call    Retardo_20ms        ; Con este retardo evitamos los rebotes del pulsador.
        btfsc   PulsadorActivacion
        goto    Activacion_Disparo
        call    LCD_Borra
        movlw   MensajeMidiendo
        call    LCD_Mensaje
        call    Retardo_2s
        bsf     Disparo             ; Comienza el pulso de disparo.
        call    Retardo_20micros    ; Duración del pulso.
        bcf     Disparo             ; Final del pulso de disparo.
```

```
Comprobacion_Funcionamiento_Sensor          ; Comprueba que el sensor funciona correctamente mediante
        decfsz  ContadorVerificarSensor,1   ; el pulso de Eco, si no hay inicio de pulso de Eco el LCD
        goto    Espera_Inicio_Eco           ; mostrará un mensaje de error.
        goto    Error_Funcionamiento_Sensor
Espera_Inicio_Eco
        btfss   Eco                 ; Si ECO=0, espera el flanco de subida de la señal de la salida del sensor.
        goto    Comprobacion_Funcionamiento_Sensor
        movlw   TMR0_Carga55micros          ; Ya se ha producido el flanco de subida.
        movwf   TMR0                        ; Carga el Timer 0.
        movlw   b'10100000'                 ; Autoriza interrupción del TMR0 debido a la activación del flag T0IE
                                            ; del registro INTCON, y así comenzar a medir la anchura del pulso.
        movwf   INTCON
Espera_Final_Eco
        btfsc   Eco                 ; Espera flanco de bajada de la señal de la salida
        goto    Espera_Final_Eco    ; de Eco del sensor.
        clrf    INTCON              ; Se ha producido el flanco de bajada. Prohíbe interrupción para dejar de
                                    ; medir el pulso de Eco.
        call    Comprobar_La_Medida         ; Comprueba la distancia y la visualiza.
        call    Retardo_2s                  ; Espera un tiempo hasta la próxima medida.
Fin     goto    Principal



; Subrutina "Servicio de Interrupcion" -----------------------------------------------------
;
; Se ejecuta debido a la petición de interrupción del Timer 0 cada 55 us que es el incremento
; de la anchura de pulso por centímetro de distancia medido. La variable "Distancia" contiene el
; valor de la distancia expresada en centímetros.
;
Interrupcion_Timer0_55us
        movlw   TMR0_Carga55micros          ; Carga el Timer 0.
        movwf   TMR0                        ; Por cada pulso de 55us incrementamos el registro de trabajo para
                                            ; calcular la duración del pulso de Eco.
        movlw   .1                          ; Utilizamos la instrucción de suma para posicionar el bit de Carry.
        addwf   Distancia,F                 ; (Distancia)+(W) = Distancia.
        movlw   MAXIMA_DISTANCIA            ; Suponemos que hay desbordamiento y cargamos su máximo valor
                                            ; (250 cm).
        btfsc   STATUS,C            ; Comparamos las distancias mediante el bit de Carry.
        movwf   Distancia           ; Si bit de Carry=1, hay desbordamiento y por lo tanto la distancia es máxima.
Final_de_la_Interrupcion
        bcf     INTCON,T0IF         ; Ponemos a 0 el flag T0IF del registro INTCON para inicializar el TMR0.
        retfie                      ; Retorna y rehabilita la interrupción (GIE=1).
```

```
; Subrutina "Error_Funcionamiento_Sensor" ----------------------------------------------------------------
;
; Indicamos el no funcionamiento del sensor mediante mensajes en el LCD. Para saber si hay error o no, hemos creado un registro (que hará de contador)
; con un valor de 250, sabiendo que entre el fin de pulso de disparo e inicio de pulso de Eco hay como mínimo una duración de 200 micro segundos.
; Por lo tanto cada bit cargado en el registro correspondrán a 1us. Si el inicio del pulso de Eco es mayor a 250us, significa que el sensor no está
; midiendo y por lo tanto el LCD mostrará un mensaje de error.
;
Error_Funcionamiento_Sensor                ; Mostramos por el LCD los mensajes correspondientes a
        call    LCD_Borra                  ; esta subrutina.
        movlw   MensajeErrorSensor
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeErrorSensor2
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MensajeComprobarFuncionamiento
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeComprobarFuncionamiento2
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        call    Retardo_2s
        goto    Principal

; Subrutina "Comprobar_La_Medida" ------------------------------------------------------------------
;
; Comprobamos la medida obtenida del pulso de Eco (a través del Timer0), comparándola con la distancia
; mínima y máxima (4 cm y 250 cm). Cuando la distancia medida sea menor de 4 cm y mayor de 250 cm, aparecerá
; un mensaje de error.
; Visualizaremos la distancia en centímetros. Se hará mediante una conversión del valor de la distancia obtenida
; en binario a BCD para poder así mostrarla en en LCD (el LCD ya incorpora un convertidor BCD-7SEGMENTS).
; Cuando haya que visualizar un número mayor de 99, las decenas siempre se visualicen aunque sean cero.
; Y cuando sea menor de 99 las decenas no se visualicen si es cero.
;
;
Comprobar_La_Medida
        call    LCD_Borra               ; Borra la pantalla anterior.
        movlw   MINIMA_DISTANCIA        ; Se comprueba si la distancia es menor del mínimo admisible.
        subwf   Distancia,W             ; (W) = (Distancia) - MINIMA_DISTANCIA
        btfss   STATUS,C                ; Si el bit de Carry=1, la distancia es mayor que la mínima
        goto    DistanciaMenor          ; Si el bit de Carry=0, la distancia es menor a la mínima y salta a
                                        ; DistanciaMenor
        movlw   MAXIMA_DISTANCIA        ; Va a comprobar si es mayor del máximo admisible.
        subwf   Distancia,W             ; (W) = (Distancia)- MAXIMA_DISTANCIA
        btfss   STATUS,C                ; Si el bit de Carry=1, la distancia es mayor a la máxima.
        goto    DistanciaFiable         ; Si el bit de Carry=0, la distancia entra dentro del rango.

DistanciaMayor
        movlw   MensajeDistancia2       ; Visualizamos que la medida está fuera del rango.
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeFueraRango
        call    LCD_Mensaje
```

```
        call    Retardo_2s
        call    LCD_Borra
        movlw   MAXIMA_DISTANCIA        ; La distancia es mayor que el máximo.
        movlw   MensajeDistanciaMayor
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeDistanciaMayor2
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        call    Retardo_1s
        goto    Principal
DistanciaMenor
        movlw   MensajeDistancia2       ; Visualizamos que la medida está fuera del rango.
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeFueraRango
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MINIMA_DISTANCIA        ; La distancia es menor del mínimo fiable.
        movlw   MensajeDistanciaMenor
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeDistanciaMenor2
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        call    Retardo_1s
        goto    Principal
DistanciaFiable
        movlw   MensajeDistancia2       ; Visualizamos que la medida está dentro del rango.
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajeDentroRango
        call    LCD_Mensaje
        call    Retardo_2s
        call    LCD_Borra
        movlw   MensajeDistancia
Visualizar_La_Distancia
        call    LCD_Mensaje
        movlw   .7                      ;Centramos la primera línea a la posición siete para que la medida
                                        ; se muestre detrás de “Dist.: “
        call    LCD_PosicionLinea1
        movf    Distancia,W
        call    BIN_a_BCD               ; Lo pasa a BCD.
        movf    BCD_Centenas,W          ; Primero las centenas.
        btfss   STATUS,Z                ; Si son cero no visualiza las centenas.
        goto    Visualiza_Centenas
        movf    Distancia,W             ; Vuelve a recuperar este valor.
        call    BIN_a_BCD               ; Lo pasa a BCD.
        call    LCD_Byte                ; Visualiza las decenas y unidades.
        goto    Visualiza_cm
Visualiza_Centenas
        call    LCD_Nibble              ; Visualiza las centenas.
        movf    Distancia,W             ; Vuelve a recuperar este valor.
        call    BIN_a_BCD               ; Lo pasa a BCD.
        call    LCD_ByteCompleto        ; Visualiza las decenas (aunque sea cero) y las unidades
```

```
Visualiza_cm
        movlw   MensajeCentimetro
        call    LCD_Mensaje
        call    LCD_PosicionLinea2
        movlw   MensajePulsarParaContinuar
        call    LCD_Mensaje
Activacion_Disparo_Para_Volver_a_medir
        btfsc   PulsadorActivacion          ; Hasta que no hay un 1 en la entrada de activación, el micro no
                                            ; disparará la señal de disparo.
        goto    Activacion_Disparo_Para_Volver_a_medir
        call    Retardo_20ms                ; Con este retardo evitamos los rebotes del pulsador.
        btfsc   PulsadorActivacion
        goto    Activacion_Disparo_Para_Volver_a_medir
        call    LCD_Borra
        return

        INCLUDE <RETARDOS_Medidor_Distancias_por_Ultrasonidos.INC>  ; Añadimos las librerías que
                                                                    ; necesitamos para completar el
        INCLUDE <LCD_4BIT_Medidor_Distancias_por_Ultrasonidos.INC>  ; programa. Dichas librerías contienen
                                                                    ; las subrutinas que hacen referencia
        INCLUDE <LCD_MENSAJES_Medidor_Distancias_por_Ultrasonidos.INC>  ; a los retardos, control en el LCD
                                                                     ; y conversión de un número binario
        INCLUDE <BIN_BCD_Medidor_Distancias_por_Ultrasonidos.INC>   ; natural a BCD.
        END
```

Librería para el LCD (Control)

```
;************** Librería "LCD_4BIT_Medidor_Distancias.INC" ********
;
; Estas subrutinas permiten realizar las tareas básicas de control de un módulo LCD de 2
; líneas por 16 caracteres, compatible con cualquier LCD que tenga el procesador Hitachi ; ; ; : ;
; 44780.
;
; El visualizador LCD está conectado al Puerto B del PIC mediante un bus de 4 bits. Las
; conexiones son:
;          - Las 4 líneas superiores del módulo LCD, pines <DB7:DB4>  se conectan a las 4
;            líneas superiores del Puerto B del PIC, pines <RB7:RB4>.
;           - Pin RS del LCD a la línea RA0 del PIC.
;           - Pin R/W del LCD a la línea RB3 del PIC, aunque lo activaremos a 0, ya que ;
;             utilizaremos el LCD sólo para visualizar datos,
;            por lo tanto en modo de escritura.
;           - Pin Enable del LCD a la línea RA1 del PIC.
;
; Se utilizan llamadas a subrutinas de retardo de tiempo localizadas en la librería
; RETARDOS_Medidor_de_distancias.INC.
;
; ZONA DE DATOS ************************

        CBLOCK
        LCD_Dato            ; Registros adiconales que utilizamos en esta libreria,
        LCD_GuardaDato      ; también quedan localizados en las direcciones de RAM
        LCD_GuardaTRISB     ; disponibles del PIC.
        LCD_Auxiliar1
        ENDC

LCD_CaracteresPorLinea EQU   .16      ; Número de caracteres por línea de la pantalla.

#DEFINE  LCD_PinRS     PORTA,0        ; Le asignamos al pin RA0 del PIC el pin RS del LCD.
#DEFINE  LCD_PinEnable PORTA,1        ; Le asignamos al pin RA1 del PIC el pin E del LCD.
#DEFINE  LCD_PinRW  PORTB,3           ; Le asignamos al pin RA3 del PIC el pin R/W del LCD.
#DEFINE  LCD_BusDatos PORTB           ; Le asignamos a los pines RB7:RB4 los pines
                                      ; DB7:DB4 del LCD.
; Subrutina "LCD_Inicializa" -----
;
; Inicialización del módulo LCD: Configura funciones del LCD, produce reset por software,
; borra memoria y enciende pantalla. El fabricante especifica que para garantizar la
; configuración inicial hay que hacerla de la siguiente manera:
;
LCD_Inicializa
        bsf     STATUS,RP0            ; Configura las líneas conectadas al pines RS,
        bcf     LCD_PinRW             ; E y RW.
        bcf     LCD_PinRS
        bcf     LCD_PinEnable
```

```
        bcf     STATUS,RP0
        bcf      LCD_PinRW          ; Le Indicamos al LCD que se va a escribir.
        bcf     LCD_PinEnable       ; Impide funcionamiento del LCD poniendo E=0.
        bcf     LCD_PinRS           ; Activa el Modo Comando poniendo RS=0.
        call    Retardo_20ms
        movlw   b'00110000'
        call    LCD_EscribeLCD      ; Escribe el dato en el LCD.
        call    Retardo_5ms
        movlw   b'00110000'
        call    LCD_EscribeLCD
        call    Retardo_200micros
        movlw   b'00110000'
        call    LCD_EscribeLCD
        call    Retardo_20micros
        movlw   b'00100000'         ; Interface de 4 bits.
        call    LCD_EscribeLCD
        call    Retardo_20micros

; Ahora configuramos el resto de parámetros mediante subrutinas,
; asignándoles la configuración adecuada para realizar la acción que nos interesa:

        call    LCD_2Lineas4Bits5x7 ; LCD de 2 líneas y caracteres de 5x7 puntos.
        call    LCD_Borra       ; Pantalla encendida y limpia. Cursor al principio de la línea 1.
        call    LCD_CursorOFF       ; Cursor apagado.
        call    LCD_CursorIncr      ; Cursor en modo incrementar.
        return

; Subrutina "LCD_EscribeLCD" ---------------------------------------------------------
;
; Envía el dato del registro de trabajo W al bus de dato y produce un pequeño pulso en el pin
; Enable del LCD. Para no alterar el contenido de las líneas de la parte baja del Puerto B que
; no son utilizadas para el LCD (pines RB3:RB0), primero se lee estas líneas y después se
; vuelve a enviar este dato sin cambiarlo.

LCD_EscribeLCD
        andlw   b'11110000'         ; Se queda con el nibble alto del dato que es el
        movwf   LCD_Dato            ; que hay que enviar y lo guarda.
        movf    LCD_BusDatos,W      ; Lee la información actual de la parte baja
        andlw   b'00001111'         ; del Puerto B, que no se debe alterar.
        iorwf   LCD_Dato,F          ; Enviará la parte alta del dato de entrada
                                    ; y en la parte baja lo que había antes.
        bsf     STATUS,RP0          ; Acceso al Banco 1.
        movf    TRISB,W             ; Guarda la configuración que tenía antes TRISB.
        movwf   LCD_GuardaTRISB
        movlw   b'00001111'         ; Las 4 líneas inferiores del Puerto B se dejan
        andwf   PORTB,F             ; como estaban y las 4 superiores como salida.
        bcf     STATUS,RP0          ; Acceso al Banco 0.
        movf    LCD_Dato,W          ; Recupera el dato a enviar.
```

```
        movwf  LCD_BusDatos          ; Envía el dato al módulo LCD.
        bsf    LCD_PinEnable         ; Permite funcionamiento del LCD mediante un pequeño.
        bcf    LCD_PinEnable         ; pulso y termina impidiendo el funcionamiento del LCD.
        bsf    STATUS,RP0            ; Acceso al Banco 1. Restaura el antiguo valor en
        movf   LCD_GuardaTRISB,W     ; la configuración del Puerto B.
        movwf  TRISB
        bcf    STATUS,RP0            ; Acceso al Banco 0.
        return

; Subrutinas variadas para el control del módulo LCD ----------
;
; Los comandos que pueden ser ejecutados son:
;
LCD_CursorIncr                       ; Cursor en modo incrementar.
        movlw  b'00000110'
        goto   LCD_EnviaComando
LCD_PosicionLinea1                   ; Cursor a posición de la Línea 1, a partir de la
        iorlw  b'10000000'           ; dirección 00h de la DDRAM más el valor del
        goto   LCD_EnviaComando      ; registro W.
LCD_PosicionLinea2                   ; Cursor a posición de la Línea 2, a partir de la
        iorlw  b'11000000'           ; dirección 40h de la DDRAM más el valor del
        goto   LCD_EnviaComando      ; registro W.
LCD_CursorOFF                        ; Pantalla encendida y cursor apagado.
        movlw  b'00001100'
        goto   LCD_EnviaComando
LCD_Borra                            ; Borra toda la pantalla, memoria DDRAM y pone el
        movlw  b'00000001'           ; cursor a principio de la línea 1.
        goto   LCD_EnviaComando
LCD_2Lineas4Bits5x7                  ; Define la pantalla de 2 líneas, con caracteres
        movlw  b'00101000'           ; de 5x7 puntos y conexión al PIC mediante bus de
        goto   LCD_EnviaComando      ; 4 bits.

; Subrutinas "LCD_EnviaComando" y "LCD_Caracter" ---------------------------------
;
; "LCD_EnviaComando". Escribe un comando en el registro del módulo LCD. La palabra de
; comando ha sido entregada a través del registro W.  Trabaja en Modo Comando.
; "LCD_Caracter". Escribe en la memoria DDRAM del LCD el carácter ASCII introducido a
; a través del registro W. Trabaja en Modo Dato.
;
LCD_EnviaComando
        bcf    LCD_PinRS       ; Activa el Modo Comando, poniendo RS=0.
        goto   LCD_Envia
LCD_Caracter
        bsf    LCD_PinRS       ; Activa el "Modo Dato", poniendo RS=1.
        movwf   LCD_Dato       ; Guarda el valor del carácter.
LCD_Envia
        movwf  LCD_GuardaDato        ; Guarda el dato a enviar.
        call   LCD_EscribeLCD        ; Primero envía el nibble alto.
```

```
        swapf   LCD_GuardaDato,W    ; Ahora envía el nibble bajo. Para ello pasa el
        call    LCD_EscribeLCD      ; nibble bajo del dato a enviar a parte alta del byte.
        btfss   LCD_PinRS           ; Se envía al visualizador LCD.
        call    Retardo_2ms         ; Debe garantizar una correcta escritura manteniendo
        call    Retardo_50micros    ; 2 ms en modo comando y 50 µs en modo carácter.
        return

; Subrutinas "LCD_Byte" y "LCD_ByteCompleto" ----------------------------------------
;
; Subrutina "LCD_ByteCompleto" visualiza los ocho bits que almacena el registro W en el lugar
; actual de la pantalla. Y si el nibble alto es cero entonces visualizará en su lugar un espacio en
; blanco.
;
; Subrutina "LCD_ByteCompleto" es igual que la anterior, pero visualiza los ocho bits
; que almacena el registro W en el lugar actual de la pantalla, sin importar
; que el nibble alto sea cero o diferente a cero.
;
LCD_Byte
        movwf  LCD_Auxiliar1        ; Guarda el valor de entrada.
        andlw  b'11110000'          ; Analiza si el nibble alto es cero.
        btfss  STATUS,Z             ; Si es cero lo apaga.
        goto   LCD_VisualizaAlto    ; No es cero y lo visualiza.
        movlw  ' '                  ; Visualiza un espacio en blanco.
        call   LCD_Caracter
        goto   LCD_VisualizaBajo
LCD_ByteCompleto
        movwf LCD_Auxiliar1         ; Guarda el valor de entrada.
LCD_VisualizaAlto
        swapf  LCD_Auxiliar1,W      ; Pone el nibble alto en la parte baja.
        call   LCD_Nibble           ; Lo visualiza.
LCD_VisualizaBajo
        movf   LCD_Auxiliar1,W      ; Repite el proceso con el nibble bajo.
        call   LCD_Nibble           ; Lo visualiza.
        return

; Subrutina "LCD_Nibble" ------------------------------------------------------------
;
; Visualiza en el lugar actual de la pantalla, el valor hexadecimal que almacena en el nibble
; bajo del registro W. El nibble alto de W no es tenido en cuenta ya que al pasar el número a
BCD
; sólo nos interesa el nibble bajo.
;
;
LCD_Nibble
        andlw  b'00001111'  ; Se queda con la parte baja.
        addlw  '0'          ; El número se pasa a carácter ASCII sumándole
        goto   LCD_Caracter ; el ASCII del cero y lo visualiza.
```

Librería para el LCD (Mensajes)

```
;********** Librería "LCD_MENSAJES_Medidor_Distancias.INC" *********
;
; Librería de subrutinas para el manejo de mensajes a visualizar en un visualizador LCD.

        CBLOCK
        LCD_ApuntaCaracter    ; Indica la posición del carácter a visualizar
                              ; respecto del comienzo de todos los mensajes,
                              ; (posición de la etiqueta "Mensajes").
        LCD_ValorCaracter     ; Código ASCII del carácter a
        ENDC                  ; visualizar.

; Los mensajes tienen que estar situados dentro de las 256 primeras posiciones de la
; memoria de programa, es decir, no pueden superar la dirección 0FFh.

LCD_Mensaje
        movwf  LCD_ApuntaCaracter          ; Posición del primer carácter del mensaje.
        movlw  Mensajes                    ; Halla la posición relativa del primer carácter
        subwf  LCD_ApuntaCaracter,F        ; del mensaje respecto de etiqueta "Mensajes".
        decf   LCD_ApuntaCaracter,F      ; Compensa la posición que ocupa "addwf PCL,F".
LCD_VisualizaOtroCaracter
        movf   LCD_ApuntaCaracter,W
        call   Mensajes              ; Obtiene el código ASCII del carácter apuntado.
        movwf  LCD_ValorCaracter     ; Guarda el valor de carácter.
        movf   LCD_ValorCaracter,F   ; Lo único que hace es posicionar flag Z. En caso
        btfsc  STATUS,Z              ; que sea "0x00", que es código indicador final
        goto   LCD_FinMensaje        ; de mensaje, sale fuera.
LCD_NoUltimoCaracter
        call   LCD_Caracter                ; Visualiza el carácter ASCII leído.
        incf   LCD_ApuntaCaracter,F        ; Apunta a la posición del siguiente carácter
        goto   LCD_VisualizaOtroCaracter   ; dentro del mensaje.
LCD_FinMensaje
        return                             ; Vuelve al programa principal.
```

Librería para los Retardos

```
;******* Librería "RETARDOS_Medidor_Distancias_por_Ultrasonidos.INC" ********
;
; Librería con múltiples subrutinas de retardos, desde 10 microsegundos hasta 4 segundos.
; Además se pueden implementar otras subrutinas muy fácilmente.
;
; Se han calculado para un sistema microcontrolador con un PIC trabajando con un cristal
; de cuarzo a 4 MHz. Como cada ciclo máquina son 4 ciclos de reloj, resulta que cada
; ciclo máquina tarda 4 x 1/4MHz = 1 µs.
;
; En los comentarios, "cm" significa "ciclos máquina".
;
; ZONA DE DATOS
*************************************************************

        CBLOCK
        R_ContA             ; Registros para los contadores para los retardos,
        R_ContB             ; ubicados en las posiciones libres de la RAM del micro.
        R_ContC
        ENDC
;
;
; RETARDOS de 10 hasta 500 microsegundos ----------------------------------
;
Retardo_200micros           ; La llamada "call" aporta 2 ciclos máquina.
        nop                 ; Aporta 1 ciclo máquina.
        movlw  d'64'        ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto    RetardoMicros   ; Aporta 2 ciclos máquina.
Retardo_50micros            ; La llamada "call" aporta 2 ciclos máquina.
        nop                 ; Aporta 1 ciclo máquina.
        movlw  d'14'        ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto    RetardoMicros   ; Aporta 2 ciclos máquina.
Retardo_20micros            ; La llamada "call" aporta 2 ciclos máquina.
        nop                 ; Aporta 1 ciclo máquina.
        movlw  d'4'         ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto   RetardoMicros    ; Aporta 2 ciclos máquinas
Retardo_10micros            ; La llamada "call" aporta 2 ciclos máquina.
        nop                 ; Aporta 1 ciclo máquina.
        nop                 ; Aporta 1 ciclo máquina.
        nop                 ; Aporta 1 ciclo máquina.
        nop                 ; Aporta 1 ciclo máquina.
        nop                 ; Aporta 1 ciclo máquina.
        nop                 ; Aporta 1 ciclo máquina.
        return              ; El salto del retorno aporta 2 ciclos máquina.
;
; El próximo bloque "RetardoMicros" tarda (2 + 3K) ciclos máquina.
```

```
; Donde K es el valor que se carga en el registro R_ContA.
;
RetardoMicros
        movwf  R_ContA
Rmicros_Bucle
        decfsz  R_ContA,F
        goto    Rmicros_Bucle
        return
;
; En total estas subrutinas tardan:
; - Retardo_200micros:  2 + 1 + 1 + 2 + (2 + 3K) = 200 cm = 200 µs. (para K= 64 y 4 MHz).
; - Retardo_50micros:   2 + 1 + 1 + 2 + (2 + 3K) =  50 cm =  50 µs. (para K= 14 y 4 MHz).
; - Retardo_20micros:   2 + 1 + 1 + 2 + (2 + 3K) =  20 cm =  20 µs. (para K=  4 y 4 MHz).
; - Retardo_10micros:   2 + 1 + 1 + 1 + 1 + 1 + 1 + 2 = 10 cm = 10 µs. (para 4 MHz).
;
; RETARDOS de 2 ms hasta 20 ms. ------------------------------------------------
;
Retardo_20ms                    ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'20'            ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto    Retardos_ms     ; Aporta 2 ciclos máquina.
Retardo_5ms                     ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'5'             ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto    Retardos_ms     ; Aporta 2 ciclos máquina.
Retardo_2ms                     ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'2'             ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto    Retardos_ms     ; Aporta 2 ciclos máquina.
;
; El próximo bloque "Retardos_ms" tarda (2 + 4M + 4KM) ciclos máquina.
; Donde M es el valor que se carga en el registro R_ContB
; y K es el valor que se carga en el registro R_ContA.
; Por lo tanto sustituyendo K=249 y M=1 en la expresión (2 + 4M + 4KM)
; obtenemos 1002 ciclos máquina, que a 4 MHz son 1002 µs = 1 ms.
;
Retardos_ms
        movwf  R_ContB
R1ms_BucleExterno
        movlw  d'249'           ; valor de 'K' que se carga en el registro.
        movwf  R_ContA
R1ms_BucleInterno
        nop
        decfsz  R_ContA,F
        goto    R1ms_BucleInterno
        decfsz  R_ContB,F
        goto    R1ms_BucleExterno
        return
;
;En total estas subrutinas tardan:
; - Retardo_20m: 2 + 1 + 2 + (2 + 4M + 4KM) =  20007 cm =  20 ms. (M= 20 y K=249).
```

```
; - Retardo_5ms: 2 + 1 + 2 + (2 + 4M + 4KM) =   5007 cm =   5 ms. (M=  5 y K=249).
; - Retardo_2ms: 2 + 1 + 2 + (2 + 4M + 4KM) =   2007 cm =   2 ms. (M=  2 y K=249).
;
; RETARDOS de 1 segundo hastas 4 segundos ----------------------------------------------------
;
Retardo_1s                          ; La llamada "call" aporta 2 ciclos máquina.
        movlw   d'10'               ; Aporta 1 ciclo máquina. Este es el valor de "N".
        goto    Retardo_Segundos    ; Aporta 2 ciclos máquina.
Retardo_2s                          ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'20'                ; Aporta 1 ciclo máquina. Este es el valor de "N".
        goto    Retardo_Segundos    ; Aporta 2 ciclos máquina.
Retardo_4s                          ; La llamada "call" aporta 2 ciclos máquina.
        movlw   d'40'               ; Aporta 1 ciclo máquina. Este es el valor de "N".
        goto   Retardo_Segundos     ; Aporta 2 ciclos máquina.
;
; El próximo bloque "Retardo_1Decima" tarda (2 + 4M + 4MN + 4KM) ciclos máquina.
; Donde N es el valor que carga en el registro R_ContC, M es el valor
; que se carga en el registro R_ContB y K es el valor que se
; carga en el registro R_ContA.
; Por lo tanto sustituyendo K=249, M=100 y N=1 en la expresión
; (2 + 4M + 4MN + 4KM) obtenemos 100011 ciclos máquina
; que a 4 MHz son 100011 µs = 100 ms = 0,1 s = 1 décima de segundo.
;
Retardo_Segundos
        movwf  R_ContC
R1Decima_BucleExterno2
        movlw  d'100'               ; valor de 'M' que se carga en el registro.
        movwf  R_ContB
R1Decima_BucleExterno
        movlw  d'249'               ; valor de 'K' que se carga en el registro.
        movwf  R_ContA
R1Decima_BucleInterno
        nop
        decfsz  R_ContA,F
        goto    R1Decima_BucleInterno
        decfsz  R_ContB,F
        goto    R1Decima_BucleExterno
        decfsz  R_ContC,F
        goto    R1Decima_BucleExterno2
        return
;
; En total estas subrutinas tardan:
; - Retardo_4s:  2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) = 5000207 cm = 5 s.
;                (N= 40, M=100 y K=249).
; - Retardo_2s:  2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) = 2000087 cm = 2 s.
;                (N= 20, M=100 y K=249).
; - Retardo_1s:  2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) = 1000047 cm = 1 s.
;                (N= 10, M=100 y K=249).
```

Librería para la Conversión de un número Binario natural a BCD

```
;****************** Librería "BIN_BCD_Medidor_Distancias.INC" **************
;
; Un número binario natural de 8 bits es convertido a BCD. El resultado se guarda en tres
; posiciones de memorias llamadas: BCD_Centenas, BCD_Decenas y BCD_Unidades.
;
; Entrada: En el registro W el número binario natural a convertir.
; Salidas: En (BCD_Centenas), (BCD_Decenas) y (BCD_Unidades).
;          En el registro W también las decenas (nibble alto) y unidades (nibble bajo).

; Subrutina "BIN_a_BCD" ---------------------------------------------

        CBLOCK
        BCD_Centenas        ; Registros para almacenar los valores de Centenas,
        BCD_Decenas         ; Decenas y Unidades. Estos registros se ubican
        BCD_Unidades        ; en las direcciones libres de la RAM.
        ENDC
;
BIN_a_BCD
        clrf    BCD_Centenas        ; Carga los registros con el resultado inicial.
        clrf    BCD_Decenas         ; En principio las centenas y decenas a cero.
        movwf   BCD_Unidades        ; Se carga el número binario a convertir.
BCD_Resta10
        movlw   .10                 ; A las unidades se les va restando 10 en cada
        subwf   BCD_Unidades,W      ; pasada. (BCD_Unidades) -10 = (W).
        btfss   STATUS,C            ; ¿C = 1?, ¿(W) positivo?, ¿(BCD_Unidades)>=10?
        goto    BIN_BCD_Fin         ; No, es menor de 10. Se acabó.
BCD_IncrementaDecenas
        movwf   BCD_Unidades        ; Recupera lo que queda por restar.
        incf    BCD_Decenas,F       ; Incrementa las decenas y comprueba si ha llegado
        movlw   .10                 ; a 10. Lo hace mediante una resta.
        subwf   BCD_Decenas,W       ; (BCD_Decenas)-10 = (W).
        btfss   STATUS,C            ; ¿C = 1?, ¿(W) positivo?, ¿(BCD_Decenas)>=10?
        goto    BCD_Resta10         ; No. Vuelve a dar otra pasada, restándole 10
BCD_IncrementaCentenas              ; a las unidades.
        clrf    BCD_Decenas         ; Pone a cero las decenas
        incf    BCD_Centenas,F      ; e incrementa las centenas.
        goto    BCD_Resta10         ; Otra pasada: Resta 10 al número a convertir.
BIN_BCD_Fin
        swapf   BCD_Decenas,W       ; En el nibble alto de (W) también las decenas.
        addwf   BCD_Unidades,W      ; En el nibble bajo de (W) las unidades.
        return                      ; Vuelve al programa principal.
```

Librería para las Instrucciones del PIC16F84A

```
	LIST

;==========================================================================
;  MPASM PIC16F84A processor include
;
;  (c) Copyright 1999-2010 Microchip Technology, All rights reserved
;==========================================================================

	NOLIST

;==========================================================================
;  This header file defines configurations, registers, and other useful
;  bits of information for the PIC16F84A microcontroller.  These names
;  are taken to match the data sheets as closely as possible.
;
;  Note that the processor must be selected before this file is included.
;  The processor may be selected the following ways:
;
;       1. Command line switch:
;            C:\MPASM MYFILE.ASM /PIC16F84A
;       2. LIST directive in the source file
;            LIST   P=PIC16F84A
;       3. Processor Type entry in the MPASM full-screen interface
;       4. Setting the processor in the MPLAB Project Dialog
;==========================================================================

;==========================================================================
;
;       Verify Processor
;
;==========================================================================
	IFNDEF __16F84A
	  MESSG "Processor-header file mismatch.  Verify selected processor."
	ENDIF
;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                EQU  H'0000'
F                EQU  H'0001'
```

```
;----- Register Files ------------------------------------------------------
;-----Bank0-----------------
INDF            EQU  H'0000'
TMR0            EQU  H'0001'
PCL             EQU  H'0002'
STATUS          EQU  H'0003'
FSR             EQU  H'0004'
PORTA           EQU  H'0005'
PORTB           EQU  H'0006'
EEDATA          EQU  H'0008'
EEADR           EQU  H'0009'
PCLATH          EQU  H'000A'
INTCON          EQU  H'000B'

;-----Bank1-----------------
OPTION_REG      EQU  H'0081'
TRISA           EQU  H'0085'
TRISB           EQU  H'0086'
EECON1          EQU  H'0088'
EECON2          EQU  H'0089'

;----- STATUS Bits --------------------------------------------------------
C               EQU  H'0000'
DC              EQU  H'0001'
Z               EQU  H'0002'
NOT_PD          EQU  H'0003'
NOT_TO          EQU  H'0004'
IRP             EQU  H'0007'

RP0             EQU  H'0005'
RP1             EQU  H'0006'

;----- PORTA Bits ---------------------------------------------------------
RA0             EQU  H'0000'
RA1             EQU  H'0001'
RA2             EQU  H'0002'
RA3             EQU  H'0003'
RA4             EQU  H'0004'

;----- PORTB Bits ---------------------------------------------------------
RB0             EQU  H'0000'
RB1             EQU  H'0001'
RB2             EQU  H'0002'
RB3             EQU  H'0003'
RB4             EQU  H'0004'
RB5             EQU  H'0005'
RB6             EQU  H'0006'
RB7             EQU  H'0007'
```

```
;----- INTCON Bits --------------------------------------------------
RBIF            EQU  H'0000'
INTF            EQU  H'0001'
T0IF            EQU  H'0002'
RBIE            EQU  H'0003'
INTE            EQU  H'0004'
T0IE            EQU  H'0005'
EEIE            EQU  H'0006'
GIE             EQU  H'0007'

TMR0IF          EQU  H'0002'
TMR0IE          EQU  H'0005'

;----- OPTION_REG Bits ----------------------------------------------
PSA             EQU  H'0003'
T0SE            EQU  H'0004'
T0CS            EQU  H'0005'
INTEDG          EQU  H'0006'
NOT_RBPU        EQU  H'0007'

PS0             EQU  H'0000'
PS1             EQU  H'0001'
PS2             EQU  H'0002'

;----- TRISA Bits ---------------------------------------------------
TRISA0          EQU  H'0000'
TRISA1          EQU  H'0001'
TRISA2          EQU  H'0002'
TRISA3          EQU  H'0003'
TRISA4          EQU  H'0004'

;----- TRISB Bits ---------------------------------------------------
TRISB0          EQU  H'0000'
TRISB1          EQU  H'0001'
TRISB2          EQU  H'0002'
TRISB3          EQU  H'0003'
TRISB4          EQU  H'0004'
TRISB5          EQU  H'0005'
TRISB6          EQU  H'0006'
TRISB7          EQU  H'0007'

;----- EECON1 Bits --------------------------------------------------
RD              EQU  H'0000'
WR              EQU  H'0001'
WREN            EQU  H'0002'
WRERR           EQU  H'0003'
EEIF            EQU  H'0004'
```

```
;==========================================================================
;
;       RAM Definitions
;
;==========================================================================
        __MAXRAM        H'00CF'
        __BADRAM        H'0007'
        __BADRAM        H'0050'-H'007F'
        __BADRAM        H'0087'
;==========================================================================
;
;       Configuration Bits
;
;   NAME        Address
;   CONFIG        2007h
;
;==========================================================================

; The following is an assignment of address values for all of the
; configuration registers for the purpose of table reads
_CONFIG         EQU  H'2007'

;----- CONFIG Options --------------------------------------------------
_LP_OSC                 EQU  H'3FFC'    ; LP oscillator
_XT_OSC                 EQU  H'3FFD'    ; XT oscillator
_HS_OSC                 EQU  H'3FFE'    ; HS oscillator
_RC_OSC                 EQU  H'3FFF'    ; RC oscillator

_WDT_OFF                EQU  H'3FFB'    ; WDT disabled
_WDT_ON                 EQU  H'3FFF'    ; WDT enabled

_PWRTE_ON               EQU  H'3FF7'    ; Power-up Timer is enabled
_PWRTE_OFF              EQU  H'3FFF'    ; Power-up Timer is disabled

_CP_ON                  EQU  H'000F'    ; All program memory is code protected
_CP_OFF                 EQU  H'3FFF'    ; Code protection disabled

;----- DEVID Equates ---------------------------------------------------
_DEVID1                 EQU  H'2006'

;----- IDLOC Equates ---------------------------------------------------
_IDLOC0                 EQU  H'2000'
_IDLOC1                 EQU  H'2001'
_IDLOC2                 EQU  H'2002'
_IDLOC3                 EQU  H'2003'

        LIST
```

ANEXO II

(Datasheets)

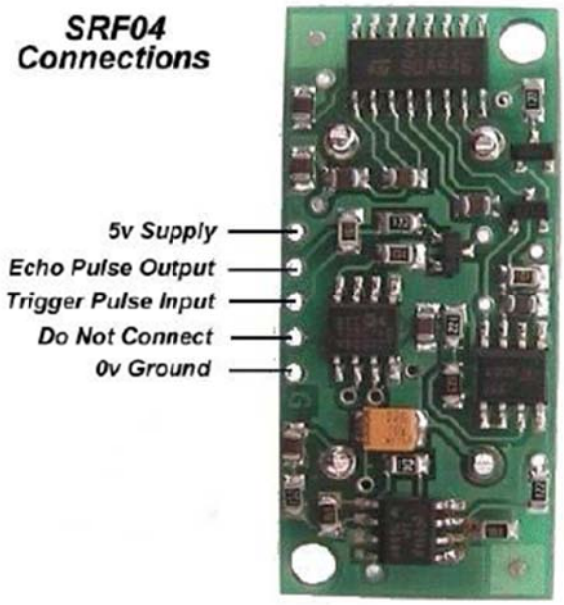
SRF04 - Ultra-Sonic Ranger

Technical Specification

This project started after I looked at the Polaroid Ultrasonic Ranging module. It has a number of disadvantages for use in small robots etc.

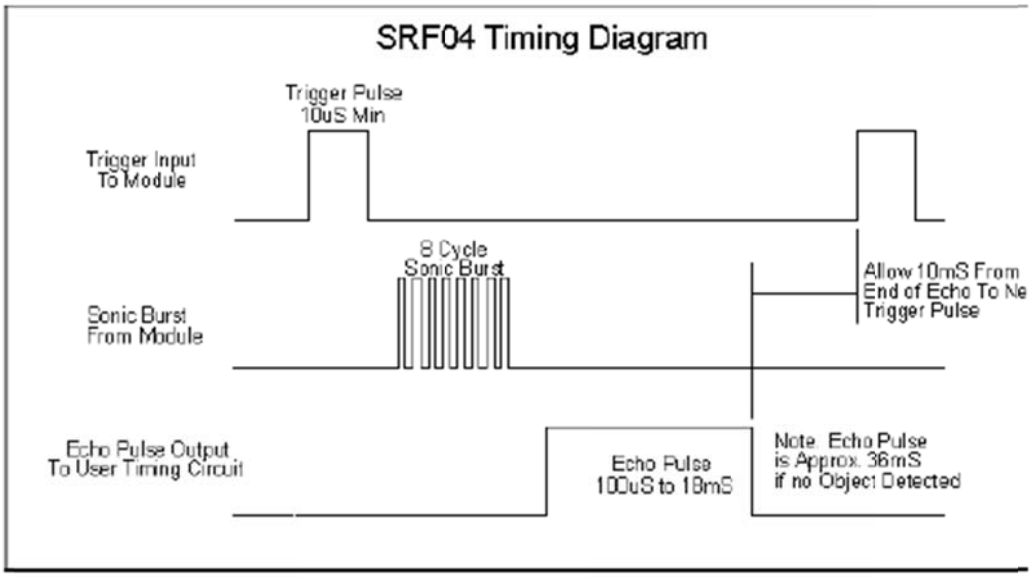
1. The maximum range of 10.7 metre is far more than is normally required, and as a result
2. The current consumption, at 2.5 Amps during the sonic burst is truly horrendous.
3. The 150mA quiescent current is also far too high.
4. The minimum range of 26cm is useless. 1-2cm is more like it.
5. The module is quite large to fit into small systems, and
6. It's EXPENSIVE.

The SRF04 was designed to be just as easy to use as the Polaroid sonar, requiring a short trigger pulse and providing an echo pulse. Your controller only has to time the length of this pulse to find the range. The connections to the SRF04 are shown below:

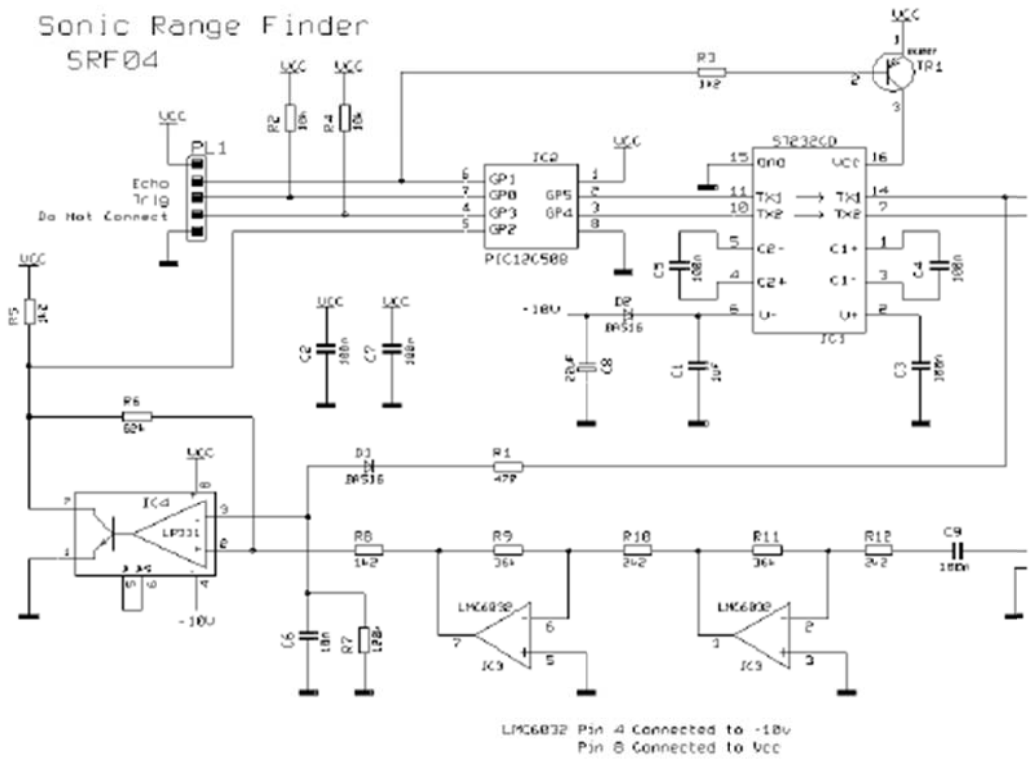


The SRF04 Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging. The SRF04 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high. It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters or anything else.

If nothing is detected then the SRF04 will lower its echo line anyway after about 36mS.



Here is the schematic, You can download a better quality pdf (161k) version srf1.pdf



The circuit is designed to be low cost. It uses a PIC12C508 to perform the control functions and

standard 40khz piezo transducers. The drive to the transmitting transducer could be simplest driven directly from the PIC. The 5v drive can give a useful range for large objects, but can be problematic detecting smaller objects. The transducer can handle 20v of drive, so I decided to get up close to this level. A MAX232 IC, usually used for RS232 communication makes and ideal driver, providing about 16v of drive.

The receiver is a classic two stage op-amp circuit. The input capacitor C8 blocks some residual DC which always seems to be present. Each gain stage is set to 24 for a total gain of 576-ish. This is close the 25 maximum gain available using the LM1458. The gain bandwidth product for the LM1458 is 1Mhz. The maximum gain at 40khz is 1000000/40000 = 25. The output of the amplifier is fed into an LM311 comparator. A small amount of positive feedback provides some hysterisis to give a clean stable output.

The problem of getting operation down to 1-2cm is that the receiver will pick up direct coupling from the transmitter, which is right next to it. To make matters worse the piezo transducer is a mechanical object that keeps resonating some time after the drive has been removed. Up to 1mS depending on when you decide it has stopped. It is much harder to tell the difference between this direct coupled ringing and a returning echo, which is why many designs, including the Polaroid module, simply blank out this period. Looking at the returning echo on an oscilloscope shows that it is much larger in magnitude at close quarters than the cross-coupled signal. I therefore adjust the detection threshold during this time so that only the echo is detectable. The 100n capacitor C10 is charged to about –6v during the burst. This discharges quite quickly through the 10k resistor R6 to restore sensitivity for more distant echo's.

A convenient negative voltage for the op-amp and comparator is generated by the MAX232. Unfortunately, this also generates quite a bit of high frequency noise. I therefore shut it down whilst listening for the echo. The 10uF capacitor C9 holds the negative rail just long enough to do this.

In operation, the processor waits for an active low trigger pulse to come in. It then generates just eight cycles of 40khz. The echo line is then raised to signal the host processor to start timing. The raising of the echo line also shuts of the MAX232. After a while – no more than 10-12mS normally, the returning echo will be detected and the PIC will lower the echo line. The width of this pulse represents the flight time of the sonic burst. If no echo is detected then it will automatically time out after about 30mS (Its two times the WDT period of the PIC). Because the MAX232 is shut down during echo detection, you must wait at least 10mS between measurement cycles for the +/- 10v to recharge.

Performance of this design is, I think, quite good. It will reliably measure down to 3cm and will continue detecting down to 1cm or less but after 2-3cm the pulse width doesn't get any smaller.

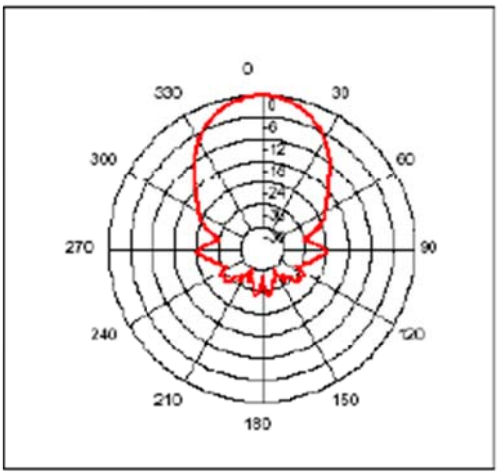
Maximum range is a little over 3m. As and example of the sensitivity of this design, it will detect a 1inch thick plastic broom handle at 2.4m.
Average current consumption is reasonable at less than 50mA and typically about 30mA.

Download the source code and a ready assembled hex file.

Calculating the Distance
The SRF04 provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches. uS/58=cm or uS/148=inches.

Changing beam pattern and beam width
You can't! This is a question which crops up regularly, however there is no easy way to reduce or change the beam width that I'm aware of. The beam pattern of the SRF04 is conical with the width of

the beam being a function of the surface area of the transducers and is fixed. The beam pattern of the transducers used on the SRF04, taken from the manufacturers data sheet, is shown below.



There is more information in the sonar faq.

Update - May 2003

Since the original design of the SRF04 was published, there have been incremental improvements to improve performance and manufacturing reliability. The op-amp is now an LMC6032 and the comparator is an LP311. The 10uF capacitor is now 22uF and a few resistor values have been tweaked. These changes have happened over a period of time.

All SRF04's manufactured after May 2003 have new software implementing an optional timing control input using the "do not connect" pin. This connection is the PIC's Vpp line used to program the chip after assembly. After programming its just an unused input with a pull-up resistor. When left unconnected the SRF04 behaves exactly as it always has and is described above. When the "do not connect" pin is connected to ground (0v), the timing is changed slightly to allow the SRF04 to work with the slower controllers such as the Picaxe. The SRF04's "do not connect" pin now acts as a timing control. **This pin is pulled high by default and when left unconnected, the timing remains exactly as before.** With the timing pin pulled low (grounded) a 300uS delay is added between the end of the trigger pulse and transmitting the sonic burst. Since the echo output is not raised until the burst is completed, there is no change to the range timing, but the 300uS delay gives the Picaxe time to sort out which pin to look at and start doing so. The new code has shipped in all SRF04's since the end of April 2003. The new code is also useful when connecting the SRF04 to the slower Stamps such as the BS2. Although the SRF04 works with the BS2, the echo line needs to be connected to the lower numbered input pins. This is because the Stamps take progressively longer to look at the higher numbered pins and can miss the rising edge of the echo signal. In this case you can connect the "do not connect" pin to ground and give it an extra 300uS to get there.

National Semiconductor

May 2000

LM78XX
Series Voltage Regulators

General Description

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and currents.

The LM78XX series is available in an aluminum TO-3 package which will allow over 1.0A load current if adequate heat sinking is provided. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistor is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating.

Considerable effort was expanded to make the LM78XX series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

For output voltage other than 5V, 12V and 15V the LM117 series provides an output voltage range from 1.2V to 57V.

Features

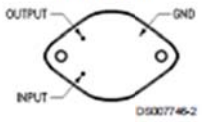
- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

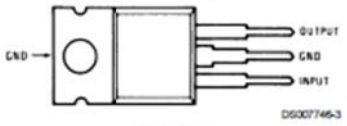
Connection Diagrams

Metal Can Package
TO-3 (K)
Aluminum



Bottom View
Order Number LM7805CK,
LM7812CK or LM7815CK
See NS Package Number KC02A

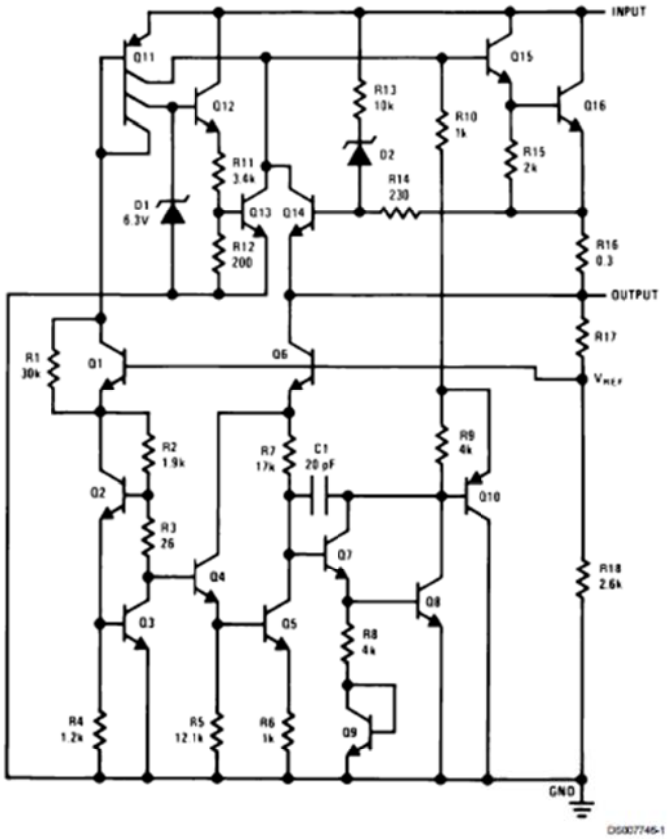
Plastic Package
TO-220 (T)



Top View
Order Number LM7805CT,
LM7812CT or LM7815CT
See NS Package Number T03B

LM78XX

Schematic



Absolute Maximum Ratings (Note 3)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Input Voltage (V_O = 5V, 12V and 15V)	35V
Internal Power Dissipation (Note 1)	Internally Limited
Operating Temperature Range (T_A)	0°C to +70°C
Maximum Junction Temperature (K Package)	150°C
(T Package)	150°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	
TO-3 Package K	300°C
TO-220 Package T	230°C

Electrical Characteristics LM78XXC (Note 2)

0°C ≤ T_J ≤ 125°C unless otherwise noted.

Output Voltage				5V			12V			15V			Units
Input Voltage (unless otherwise noted)				10V			19V			23V			
Symbol	Parameter	Conditions		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_O	Output Voltage	Tj = 25°C, 5 mA ≤ I_O ≤ 1A		4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
		P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A		4.75		5.25	11.4		12.6	14.25		15.75	V
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7.5 ≤ V_{IN} ≤ 20)			(14.5 ≤ V_{IN} ≤ 27)			(17.5 ≤ V_{IN} ≤ 30)			V
ΔV_O	Line Regulation	I_O = 500 mA	Tj = 25°C		3	50		4	120		4	150	mV
			ΔV_{IN}	(7 ≤ V_{IN} ≤ 25)			14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)			V
			0°C ≤ Tj ≤ +125°C			50			120			150	mV
			ΔV_{IN}	(8 ≤ V_{IN} ≤ 20)			(15 ≤ V_{IN} ≤ 27)			(18.5 ≤ V_{IN} ≤ 30)			V
		I_O ≤ 1A	Tj = 25°C			50			120			150	mV
			ΔV_{IN}	(7.5 ≤ V_{IN} ≤ 20)			(14.6 ≤ V_{IN} ≤ 27)			(17.7 ≤ V_{IN} ≤ 30)			V
			0°C ≤ Tj ≤ +125°C			25			60			75	mV
			ΔV_{IN}	(8 ≤ V_{IN} ≤ 12)			(16 ≤ V_{IN} ≤ 22)			(20 ≤ V_{IN} ≤ 26)			V
ΔV_O	Load Regulation	Tj = 25°C	5 mA ≤ I_O ≤ 1.5A		10	50		12	120		12	150	mV
			250 mA ≤ I_O ≤ 750 mA			25			60			75	mV
		5 mA ≤ I_O ≤ 1A, 0°C ≤ Tj ≤ +125°C				50			120			150	mV
I_Q	Quiescent Current	I_O ≤ 1A	Tj = 25°C			8			8			8	mA
			0°C ≤ Tj ≤ +125°C			8.5			8.5			8.5	mA
ΔI_Q	Quiescent Current Change	5 mA ≤ I_O ≤ 1A				0.5			0.5			0.5	mA
		Tj = 25°C, I_O ≤ 1A				1.0			1.0			1.0	mA
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN}≤ 27)			(17.9 ≤ V_{IN} ≤ 30)			V
		I_O ≤ 500 mA, 0°C ≤ Tj ≤ +125°C				1.0			1.0			1.0	mA
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7 ≤ V_{IN} ≤ 25)			(14.5 ≤ V_{IN}≤ 30)			(17.5 ≤ V_{IN} ≤ 30)			V
V_N	Output Noise Voltage	T_A =25°C, 10 Hz ≤ f ≤ 100 kHz			40			75			90		µV
$\frac{\Delta V_{IN}}{\Delta V_{OUT}}$	Ripple Rejection		I_O ≤ 1A, Tj = 25°C or	62	80		55	72		54	70		dB
		f = 120 Hz	I_O ≤ 500 mA	62			55			54			dB
			0°C ≤ Tj ≤ +125°C										
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)			V
R_O	Dropout Voltage	Tj = 25°C, I_{OUT} = 1A			2.0			2.0			2.0		V
	Output Resistance	f = 1 kHz			8			18			19		mΩ

Electrical Characteristics LM78XXC (Note 2) (Continued)

$0°C \le T_J \le 125°C$ unless otherwise noted.

Output Voltage			5V			12V			15V			Units
Input Voltage (unless otherwise noted)			10V			19V			23V			
Symbol	Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
	Short-Circuit Current	Tj = 25°C		2.1			1.5			1.2		A
	Peak Output Current	Tj = 25°C		2.4			2.4			2.4		A
	Average TC of V_{OUT}	$0°C \le Tj \le +125°C$, I_O = 5 mA		0.6			1.5			1.8		mV/°C
V_{IN}	Input Voltage Required to Maintain Line Regulation	Tj = 25°C, $I_O \le 1A$		7.5		14.6			17.7			V

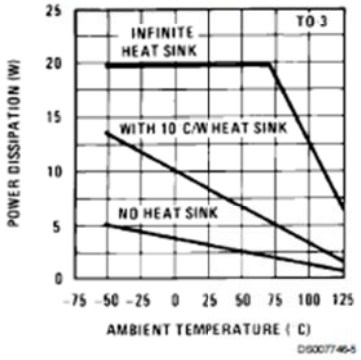
Note 1: Thermal resistance of the TO-3 package (K, KC) is typically 4°C/W junction to case and 35°C/W case to ambient. Thermal resistance of the TO-220 package (T) is typically 4°C/W junction to case and 50°C/W case to ambient.

Note 2: All characteristics are measured with capacitor across the input of 0.22 µF, and a capacitor across the output of 0.1µF. All characteristics except noise voltage and ripple rejection ratio are measured using pulse techniques ($t_w \le 10$ ms, duty cycle ≤ 5%). Output voltage changes due to changes in internal temperature must be taken into account separately.

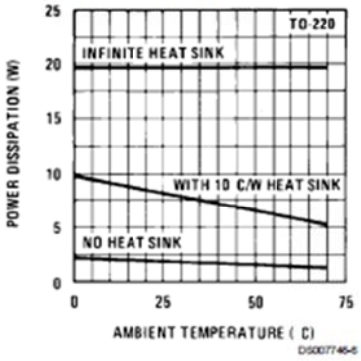
Note 3: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. For guaranteed specifications and the test conditions, see Electrical Characteristics.

Typical Performance Characteristics

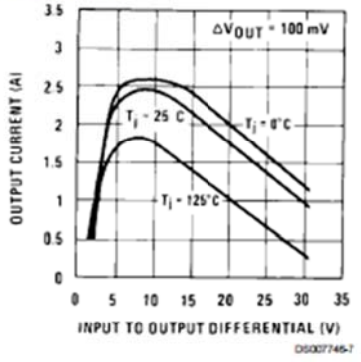
Maximum Average Power Dissipation



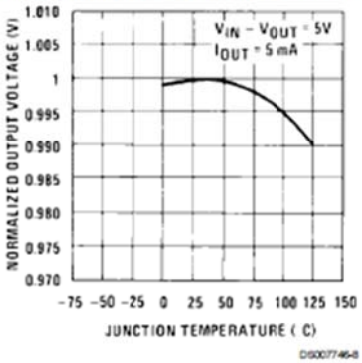
Maximum Average Power Dissipation



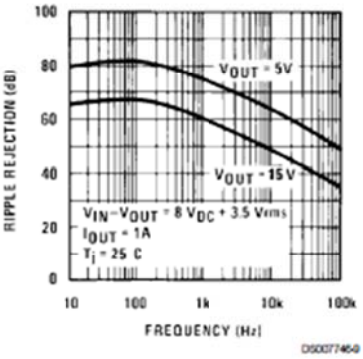
Peak Output Current



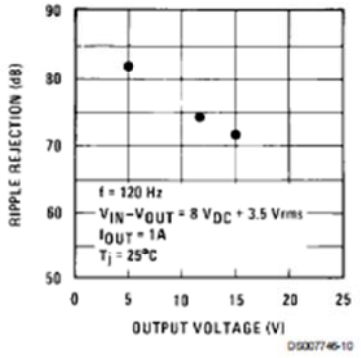
Output Voltage (Normalized to 1V at T_J = 25°C)



Ripple Rejection

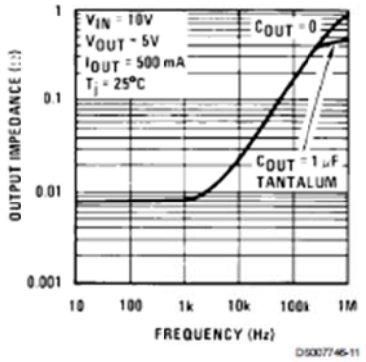


Ripple Rejection

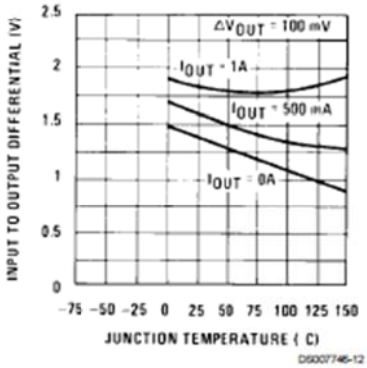


Typical Performance Characteristics (Continued)

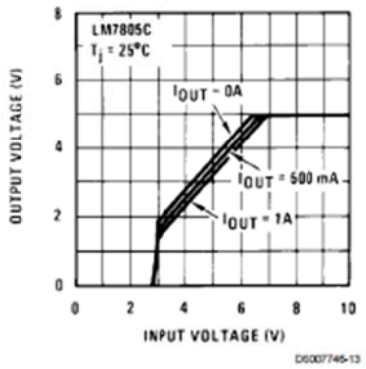
Output Impedance



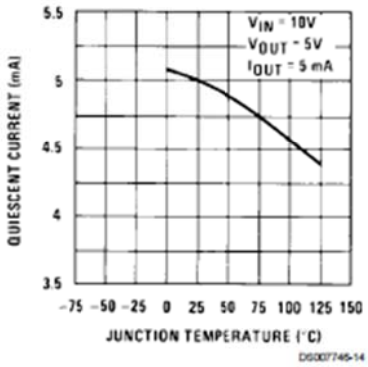
Dropout Voltage



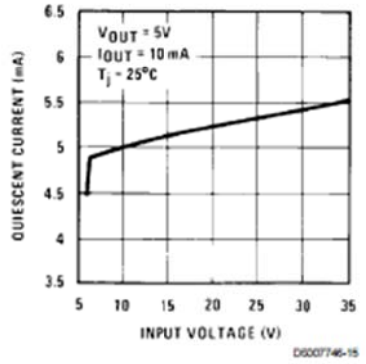
Dropout Characteristics



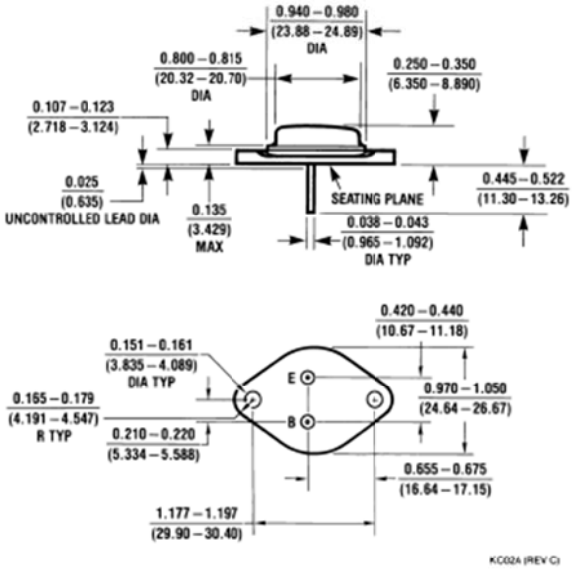
Quiescent Current



Quiescent Current

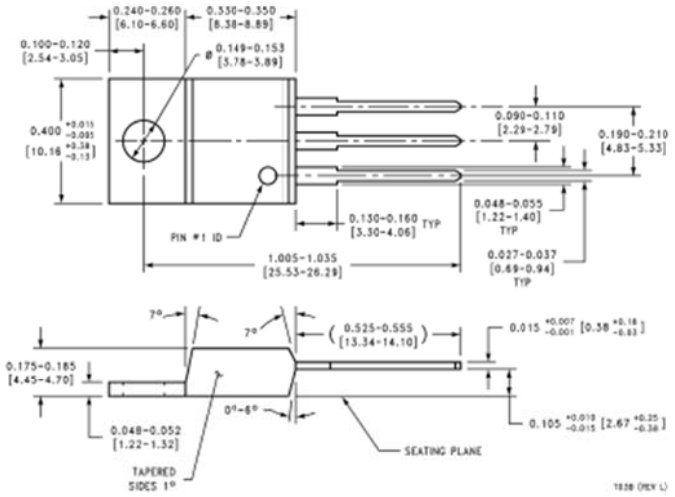


Physical Dimensions inches (millimeters) unless otherwise noted



Aluminum Metal Can Package (KC)
Order Number LM7805CK, LM7812CK or LM7815CK
NS Package Number KC02A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



TO-220 Package (T)
Order Number LM7805CT, LM7812CT or LM7815CT
NS Package Number T03B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

POWERTIP TECH. CORP.

DISPLAY DEVICES FOR BETTER ELECTRONIC DESIGN

Specification For Approval

Customer : _____

Model Type : LCD Module

Sample Code : PC1602LRS-HSO-B-S0

Mass Production Code : _____

Edition : 0

Customer Sign	Sales Sign	Approved By	Prepared By

NO.PC1602LRS-HSO-B

1. SPECIFICATIONS

1.1 Features

- 16-characters, two-lines liquid crystal display of 5*8 dot matrix + cursor
- 1/16 Duty, 1/4 bias
- STN LCD, positive, gray
- Transflective LCD
- 6 o'clock viewing angle
- 8 bits parallel data input
- Built-in LED backlight

1.2 Mechanical Specifications

- Outline dimension : 85.0mm(L)* 36.0mm(W)*14.0mm max.(H)
- Viewing area : 66.0mm *16.2mm
- Active area : 56.21mm *11.5mm
- Dot size : 0.56mm *0.66mm
- Dot pitch : 0.6mm *0.7mm
- Character Size : 2.96mm *5.56mm

1.3 Absolute Maximum Ratings

Item	Symbol	Conditions	Min.	Max.	Unit
Power supply Voltage	VDD	-	-0.3	7.0	V
LCD drive Supply voltage	VDD-Vo	-	V_{DD}-15	V_{DD}+0.3	V
Input voltage	VIN	-	-0.3	V_{DD}+0.3	V
Operating temperature	TOPR	-	0	50	°C
Storage temperature	TSTG	-	-20	+70	°C
Humidity	HD	-	-	90	%RH

1.4 DC Electrical Characteristics

V_{DD}=+5V±10%,V_{SS}=0V,T_A=25°C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Logic Supply voltage	VDD	-	4.5	5.0	5.5	V
"H" input voltage	VIH	-	2.2	-	V_{DD}	V
"L" input voltage	VIL	-	-0.3	-	0.6	V
"H" output voltage	VOH	-	2.4	-	-	V
"L" output voltage	VOL	-	-	-	0.4	V
Supply current	IDD	VDD=5V	-	1.66	1.99	mA
LCD driving voltage	VOP	VDD-Vo	-	4.4	4.8	V

POWERTIP TECHNOLOGY CORPORATION

DISPLAY DEVICES FOR BETTER ELECTRONIC DESIGN

1.5 Optical Characteristics

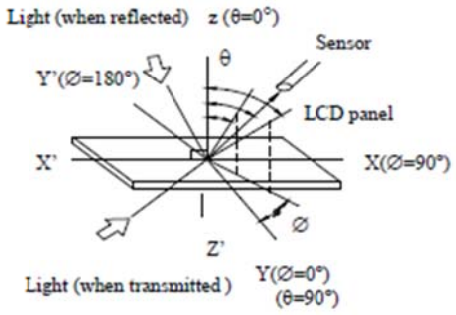
1/16 duty, 1/4 bias, V_{OPR}=4.2V, Ta=25°C

Item	Symbol	Conditions	Min.	Typ.	Max	Reference
Viewing angle	θ	C≥2.0,∅=0°	-40°	-	-	Notes 1 & 2
Contrast	C	θ=5°, ∅=0°	-	3	-	Note 3
Response time(rise)	T_r	θ=5°, ∅=0°	-	120 ms	180 ms	Note 4
Response time(fall)	T_f	θ=5°, ∅=0°	-	250 ms	400 ms	Note 4

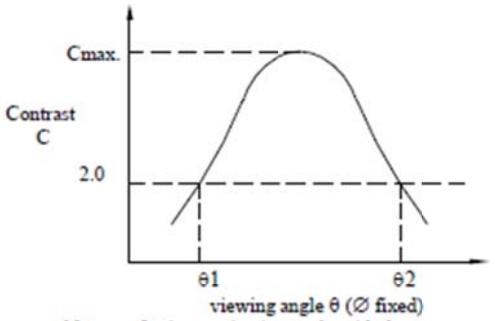
Parameter	Symbol	Temperature (°C)	Standard			Unit
			Min	Typ	Max	
Driving voltage	V_{OP}	0	4.3	4.6	4.9	V
		25	3.9	4.2	4.5	
		40	3.7	4.0	4.3	

NO.PC1602LRS-HSO-B

Note 1: Definition of angles θ and ∅



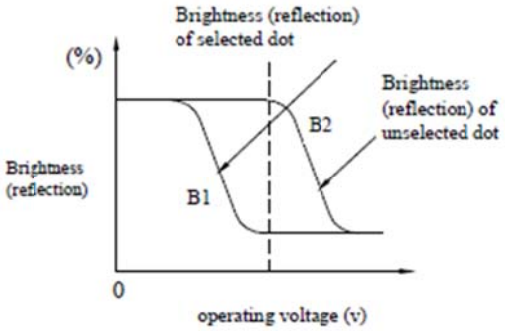
Note 2: Definition of viewing angles θ1 and θ2



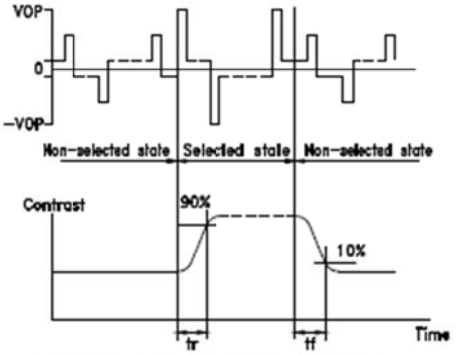
Note : Optimum viewing angle with the naked eye and viewing angle θ at Cmax. Above are not always the same

Note 3: Definition of contrast C

$$C = \frac{\text{Brightness (reflection) of unselected dot (B2)}}{\text{Brightness (reflection) of selected dot (B1)}}$$



Note 4: Definition of response time



Note: Measured with a transmissive LCD panel which is displayed 1 cm^2

V_{OPR} : Operating voltage
t_r : Response time (rise)
f_{FRM} : Frame frequency
t_f : Response time (fall)

1.6 Backlight Characteristic

The LCD Module is backlight using a LED panel

•Maximum Ratings

NO.PC1602LRS-HSO-B

Item	Symbol	Conditions	Min.	Max.	Unit
Forward current	IF	TA=25°C	-	240	mA
Reverse voltage	VR	TA=25°C	-	8	V
Power dissipation	PO	TA=25°C	-	1.2	W
Operating Temperature	TOPR	-	-20	70	°C
Storage temperature	TSTG	-	-40	80	°C

•Electrical Ratings

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Forward voltage	VF	IF=120mA	-	4.2	4.8	V
Reverse current	IR	VR=8V	-	-	0.3	mA
Luminous intensity (without LCD)	IV	IF=120mA	50	80	-	cd/m^2
Luminous intensity (with LCD)	IV	IF=120mA	-	40.2	-	cd/m^2
Wavelength	λp	IF=120mA	569	-	575	nm
Color	Yellow Green					

POWERTIP TECHNOLOGY CORPORATION

DISPLAY DEVICES FOR BETTER ELECTRONIC DESIGN

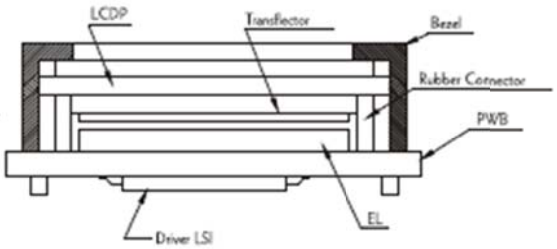
BACKLIGHT FOR LCD MODULES

EL Backlight

Flat surface light source offers simple and even illumination over large area.

Features

(1)Max.1.3mm thickness (Max. 1.5mm for lead portion)
(2)Wide driving condition, 60 - 1,000Hz at 150V AC Max
(3)With inverter, step-up voltage from 1.5V battery is available
(4)Emitted colors are blue-green, yellow-green and white
(5)Operating characteristics of PC2002-A SERIES is 110V, 400Hz, 8mA, (Ta=20℃, 60% RH)
(6)Temperature range: Operating 0℃~ +50℃ ;
Storage -20℃~ +60℃



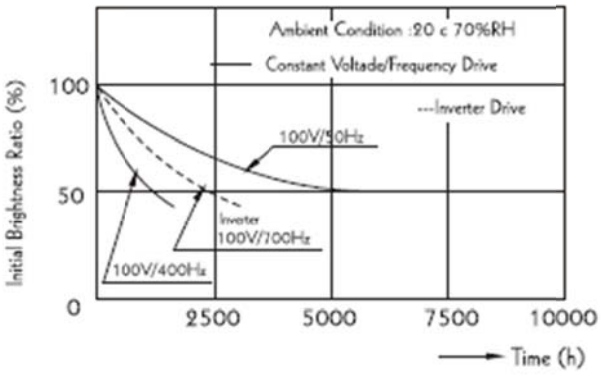
Precation

Requires an inverter to operate the EL panel with a battery or DC power supply.

(1)Low inverter loss and high light efficiency as it is designed for EL backlight

(2)Less change of power consumption during operation under temperature change or extended hours. This is characterized by the constant supply current, which minimizes the brightness change of the EL panel

Life Characteristics



CCFL Backlight (Cold Cathode Fluorescent Lamp)

Bright white color of light source offers clear and even illumination over large viewing area.

Features

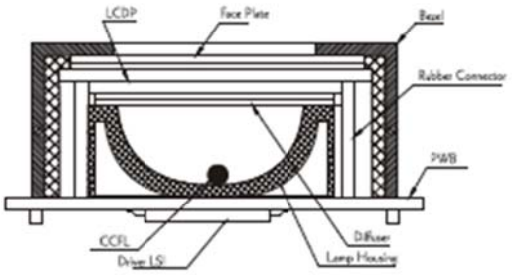
(1)High Brightness
(2)Long lifetime.
(3)White color emitted
(4)Temperature range: Operating: 0℃~ +50℃;
Storage -20℃~ +60℃
(5)Life time:10,000 ~20,000 hours

Precation

Inverter for CCFL use output high pressure AC current. Therefore, please pay attention when you handle inverter and power supply cable of LCD backlight.

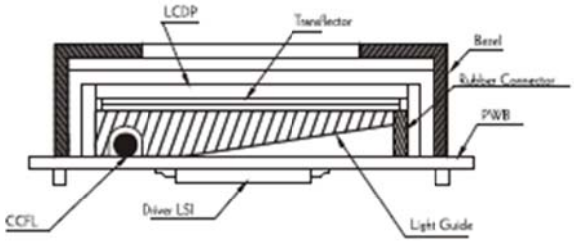
Direct Illumination

Suitable for multi-color and / or dot matrix LCD



Edge Illumination

Thin structure type of even illumination emits light from tube-like light source over a large area.



◆LED Backlight

Long life, low power consumption and requires a simple power supply. Available colors are red, green and orange, available in array type illumination or edge illumination

Features

(1)Low driving voltage (DC) and does not require an inverter

(2)Long life (100,000 hours/average)

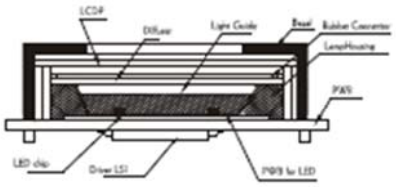
(3)No noise occurrence

(4)Various colors available in red, green and orange etc (multi-color by alternative switch is also available)

(5)Operating characteristics of PC2002-A series is 4.2V, 210mA, 250cd/m

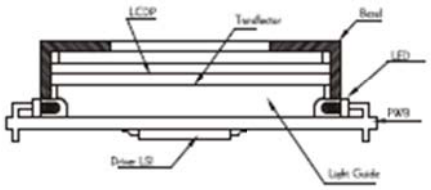
Array Illumination

A grid array of LED's provide even illumination.



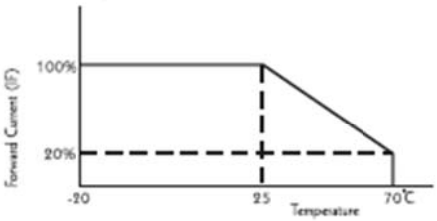
Edge Illumination

Combination LED with a light guide offers a thin structure type of illumination.

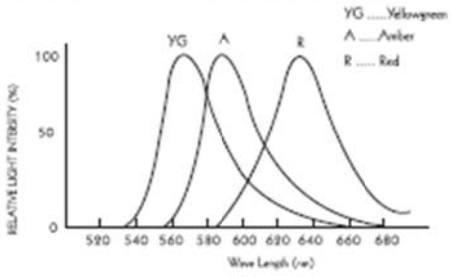


Electrical Characteristics (Reference Data)

Forward Current De-Rating Curve



Wave Length Vs Relative Light Intensity



2. MODULE STRUCTURE

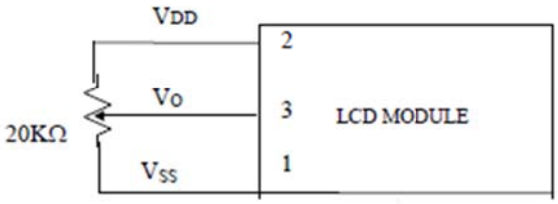
2.1 Counter Drawing

*See Appendix

2.2 Interface Pin Description

Pin No.	Symbol	Signal Description
1	VSS	Power Supply (V_{SS}=0)
2	VDD	Power Supply (V_{DD}>V_{SS})
3	VO	Operating voltage (LCD Driver)
4	RS	Register Selection input High = Data register Low = Instruction register (for write) Busy flag address counter (for read)
5	— R/W	Read/Write signal input is used to select the read/write mode High = Read mode, Low = Write mode
6	E	Start enable signal to read or write the data
7~10	DB0 ~ DB3	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCD module. These four are not used during 4-bit operation.
11~14	DB4 ~ DB7	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCD module. DB7 can be used as a busy flag.
15	A	Power supply for LED B / L (+)
16	K	Power supply for LED B / L (-)

Contrast Adjust



2.3 Timing Characteristics

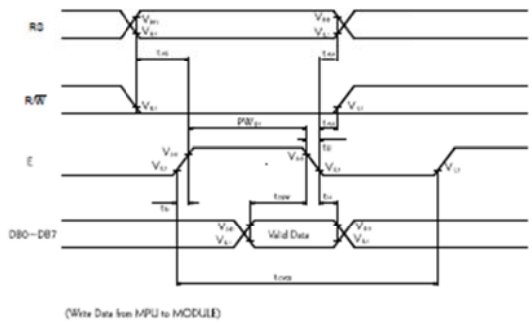
TIMING CHARACTERISTICS

Vdd = 5V ±5%
Vss = 0V

Item		Symbol	Min.	Max.	Unit.
Enable cycle time		Tcyce	500	—	ns
Enable pulse width	"High" level	PWEH	200	—	ns
Enable rise/fall time		TER, TEF	—	25	ns
Set-up time	RS,R/W,E	TAS	40	—	ns
Address hold time		TAH	10	—	ns
Data Set-up time		TDSH	60	—	ns
Data delay time		TDDR	60	120	ns
Data hold time (writing)		TH	10	—	ns
Data hold time (reading)		TDHR	20	—	ns
Clock oscillating frequency		Tosc	270(Typ.)		KHz

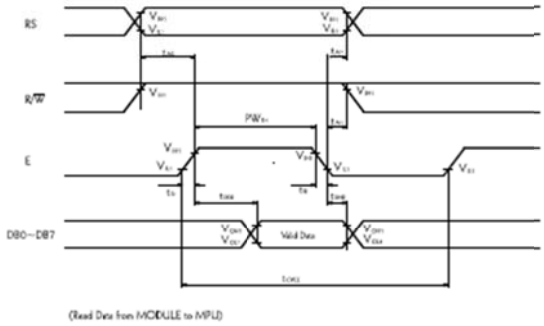
TIMING CHART

FIG.1 WRITE OPERATION



(Write Data from MPU to MODULE)

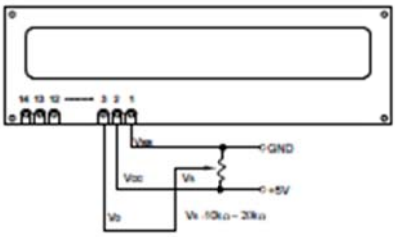
FIG.2 READ OPERATION



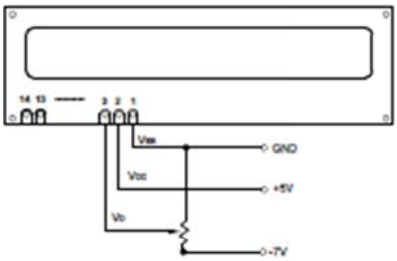
(Read Data from MODULE to MPU)

EXAMPLE OF POWER SUPPLY

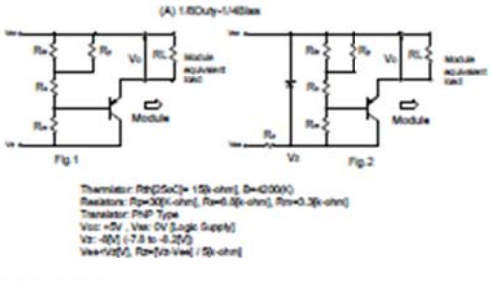
Normal Temperature Type

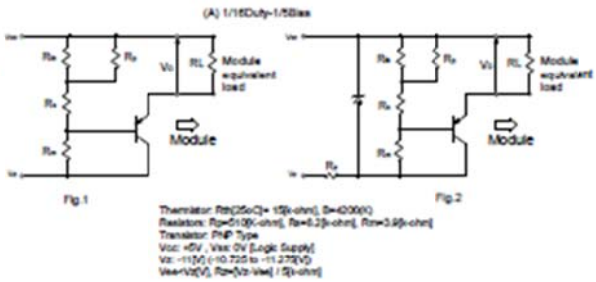


Extended Temperature Type



Examples of Temperature Compensation Circuits for Extended Temp. (Only for reference)





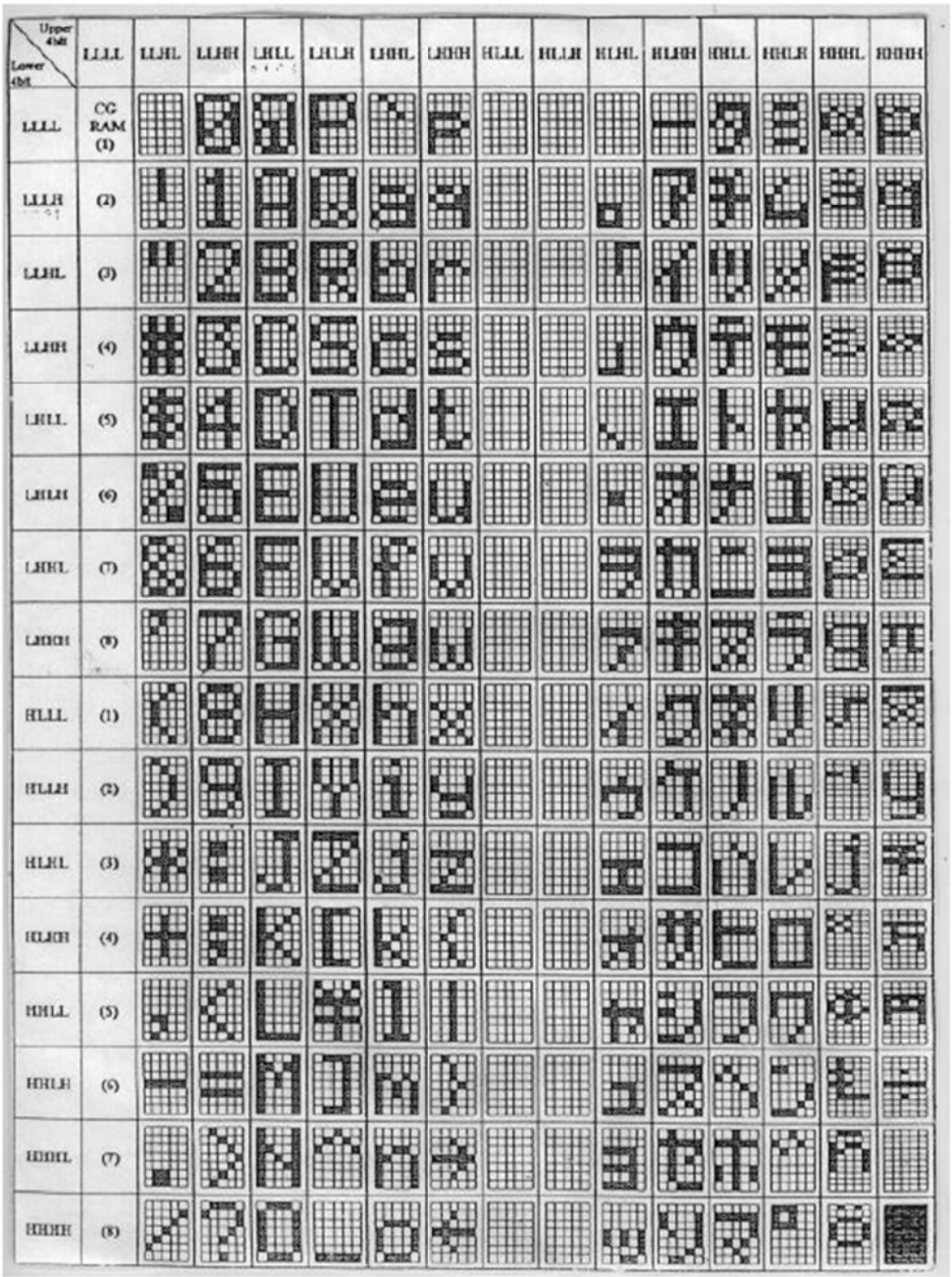
INSTRUCTIONS

Instruction	Code										Description	Executed Time(max.)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	1.64mS
Cursor Art Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home positio (Address 0). Also returns the display bing shifted to the original position. DD RAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40μS
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor NO/OFF (C), and blink of cursor position character (B).	40μS
Cursor /Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents.	40μS
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F)	40μS
CG RAM Address Set	0	0	0	1	ACG						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40μS
DD RAM Address Set	0	0	1	ADD							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40μS
Busy Flag/ Address Read	0	1	BF	AC							Reads Busy flag (FB) indicating internal operation is being performed and reads address counter counts.	0μS
CG RAM / DD RAM Data Write	1	0	WRITE DATA								Writes data into DD RAM or CG RAM.	40μS
CG RAM / DD RAM Data Read	1	1	READ DATA								Reads data from DD RAM or CG RAM.	40μS

Code		Description	Executed Time (max)
I/D = 1 : Increment I/D = 0 : Decrement S = 1 : With display shift S / C = 0 : cursor movement R / L =1 : Shift to the right R / L = 0 : Shift to the left D L = 1 : 8-bit	D / L = 0 : 4-bit N = 1 : 2 lines N = 0 : 1 line F = 1 : 5 x 10 dots F =0.5 x 7 dots BF = 1 : internal operation is being performed BF = 0 : instruction acceptable	DD RAM : Display Data RAM CG RAM : Character Generator RAM ACG : CG RAM Address ADD : DD RAM Address Corresponds to cursor address AC : Address Counter, used for both DD RAM and CG RAM * : Invalid	fcp or fosc = 250 KHz However. when frequency changes. eecution time also changes Example if fcp or fosc is 270KHz, 70μS x 250 / 270 = 37μS

NO.PC1602LRS-HSO-B

2.5 Character Pattern

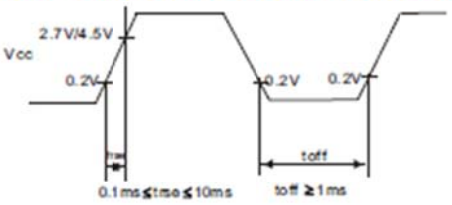


POWERTIP TECHNOLOGY CORPORATION

DISPLAY DEVICES FOR BETTER ELECTRONIC DESIGN

POWER SUPPLY RESET

The internal reset circuit will be operating properly when the foll owing power supply conditions are satisfied. If it isnot operated properly, please perform the initial setting along with the instruction.



Item	Symbol	Measuring Condition	Standard Value			Unit
			Min.	Typ.	Max.	
Power Supply RISE Time	tree	—	0.1	—	10	mS
Power Supply CFF Time	toff	—	1	—	—	mS

Reset Function

Initialization Mode by Internal Reset Circuit

The controller automatically initializes (resets) when power is supplied (built-in internal reset circuit). The following instructions are executed during initialization. The busy flag (BF) is kept in busy state until initialization ends (BF=1). The busy state is 10ms after Vdd reaches 4.5V.

(1) Display clear

(2) Function set
- DL=1:8 bit long interface data
- DL=0:4 bit F=0:5 x 7 dots character font
- N=1:2 lines
- N=0:1 line

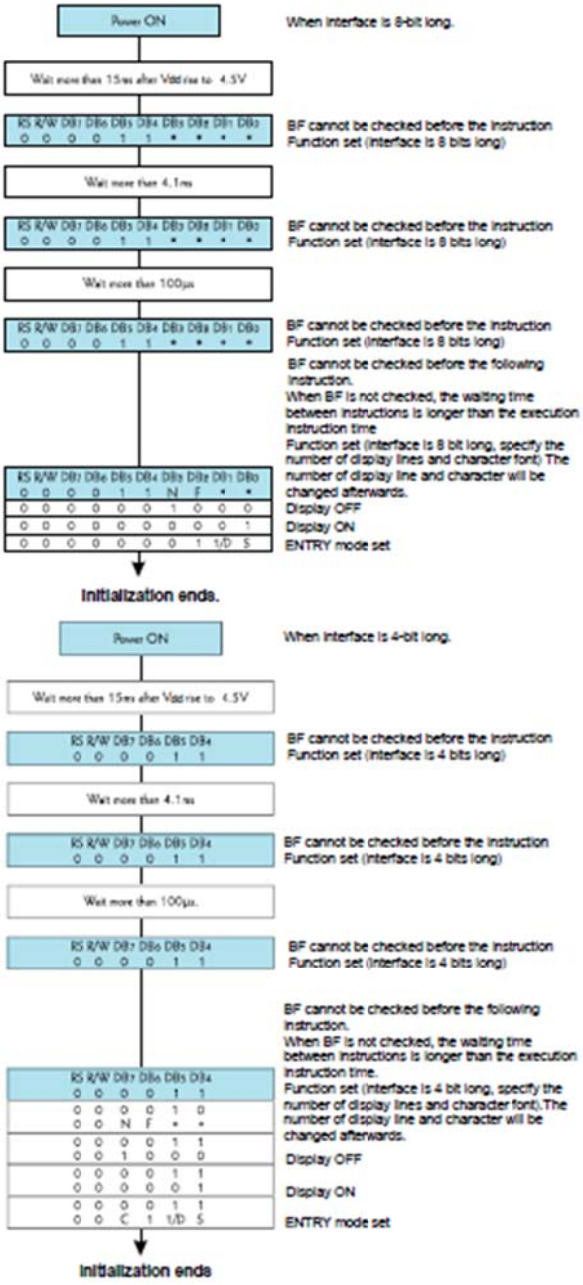
(3) Display ON/OFF control
- D=0:Display OFF
- C=0:Cursor OFF
- B=0:Blink OFF

(4) Entry mode set
- 1 / D = 1: + 1 (increment)
- S = 0: No shift

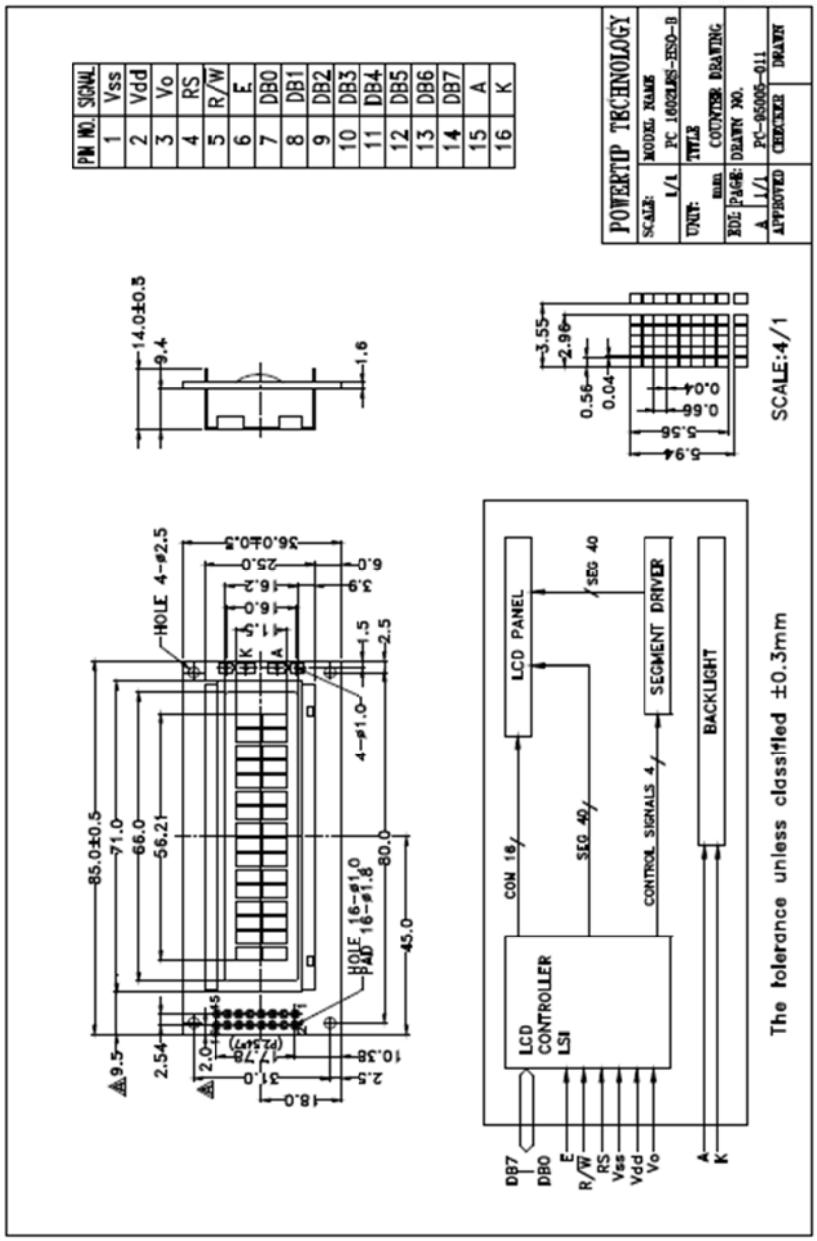
Note: When using internal reset circuit is not satisfied in power supply conditions, the internal reset circuit will not function properly and initialization will not be performed. Please initialize using the MPU along with the instrcutuion set.

Initialization along with Instruction

If power supply conditions are not satisfied, which for proper operation of internal rest circuit, it is required to make initialzation along with instruction.
Please make following procedures.



NO.PC1602LRS-HSO-B



POWERTIP TECHNOLOGY CORPORATION

DISPLAY DEVICES FOR BETTER ELECTRONIC DESIGN

PIC16F84A
Data Sheet

18-pin Enhanced FLASH/EEPROM
8-bit Microcontroller

 DS35007B

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable".
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

MICROCHIP

PIC16F84A

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

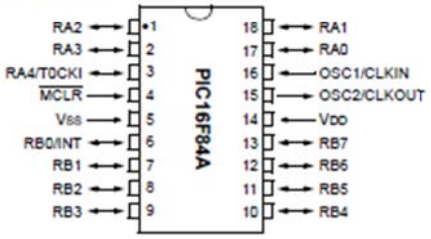
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

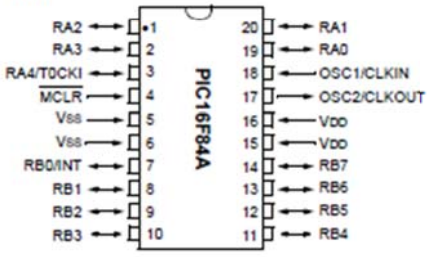
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

PIC16F84A

Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

This document contains device specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro® microcontroller devices. A block diagram of the device is shown in Figure 1-1.

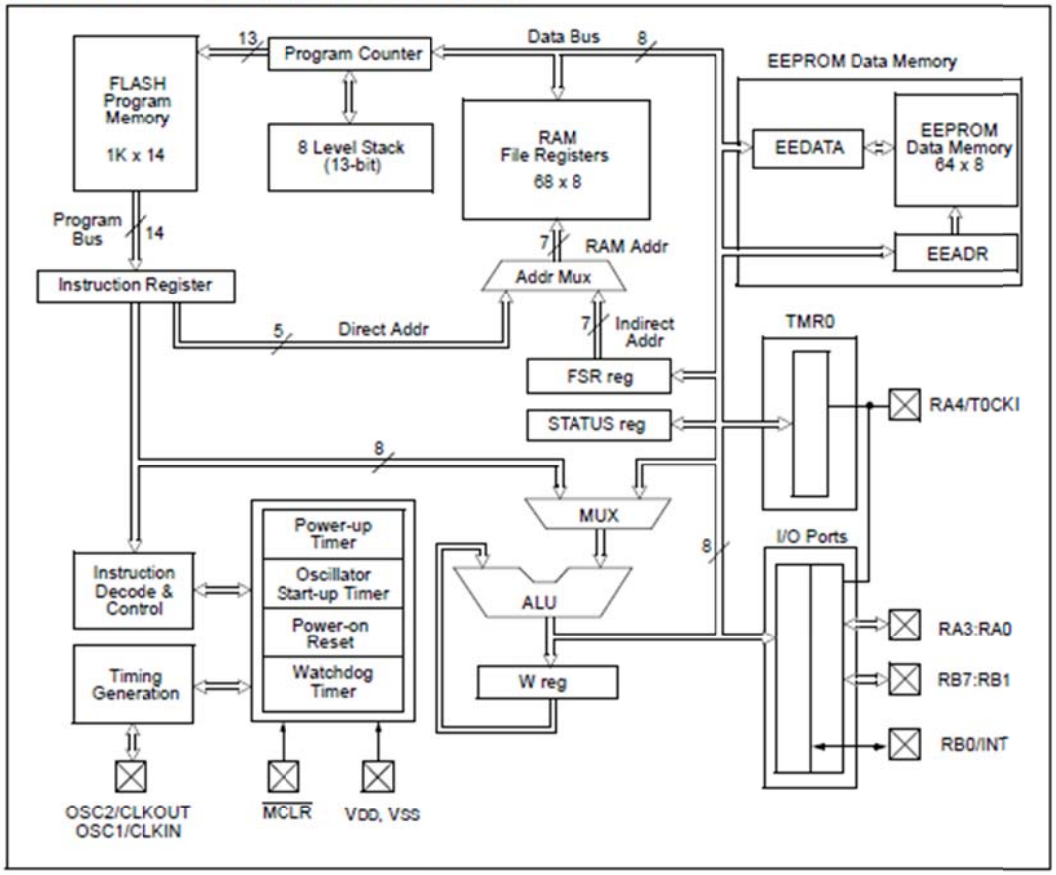
The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



PIC16F84A

TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 3.0.

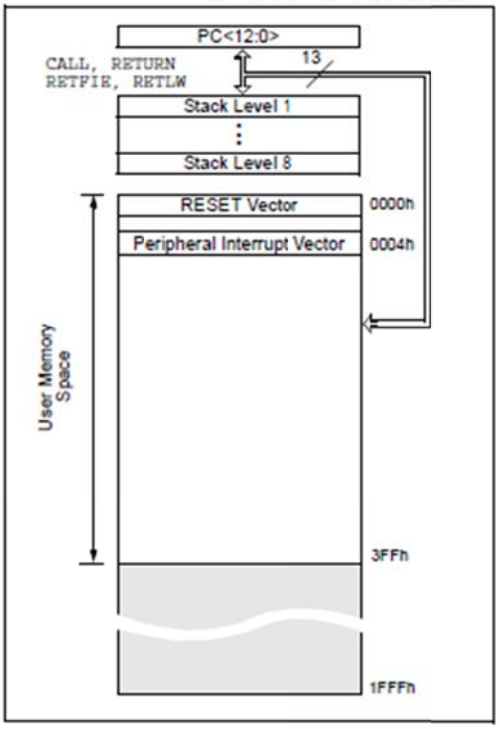
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h, the instruction will be the same.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A



PIC16F84A

2.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 2-2 shows the data memory map organization.

Instructions `MOVWF` and `MOVF` can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 2.5). Indirect addressing uses the present value of the RP0 bit for access into the banked areas of data memory.

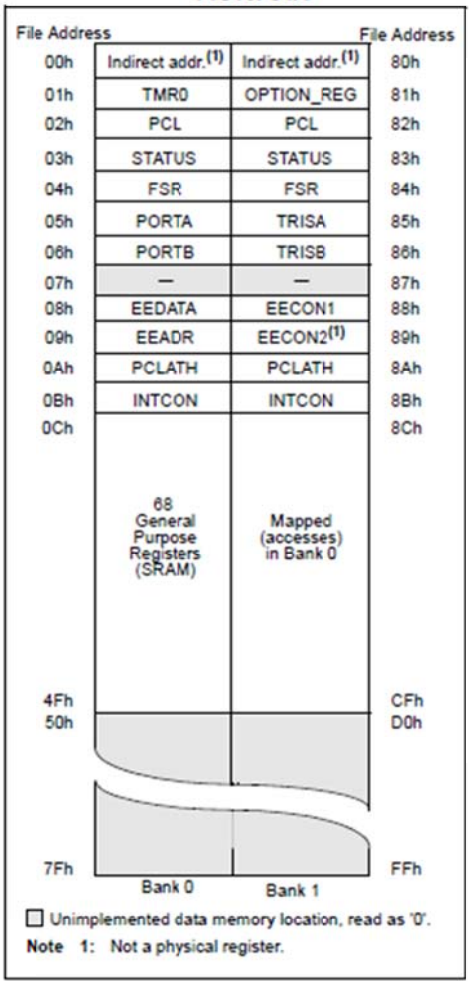
Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers, implemented as static RAM.

2.2.1 GENERAL PURPOSE REGISTER FILE

Each General Purpose Register (GPR) is 8-bits wide and is accessed either directly or indirectly through the FSR (Section 2.5).

The GPR addresses in Bank 1 are mapped to addresses in Bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



2.3 Special Function Registers

The Special Function Registers (Figure 2-2 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.

3: Other (non power-up) RESETS include: external RESET through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

PIC16F84A

2.3.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u uluu` (where `u` = unchanged).

Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 7-2), because these instructions do not affect any status bit.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

2: The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

3: When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 RP0: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 $\overline{TO}$: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 $\overline{PD}$: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 Z: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 DC: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 C: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.3.2 OPTION REGISTER

The OPTION register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

Note: When the prescaler is assigned to the WDT (PSA = '1'), TMR0 has a 1:1 prescaler assignment.

REGISTER 2-2: OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS:** TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE:** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0:** Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC16F84A

2.3.3 INTCON REGISTER

The INTCON register is a readable and writable register that contains the various enable bits for all interrupt sources.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.4 PCL and PCLATH

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. If the program counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP. All updates to the PCH register go through the PCLATH register.

2.4.1 STACK

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Mid-range devices have an 8 level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not modified when the stack is PUSHed or POPed.

After the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

2.5 Indirect Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 2-1: INDIRECT ADDRESSING

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

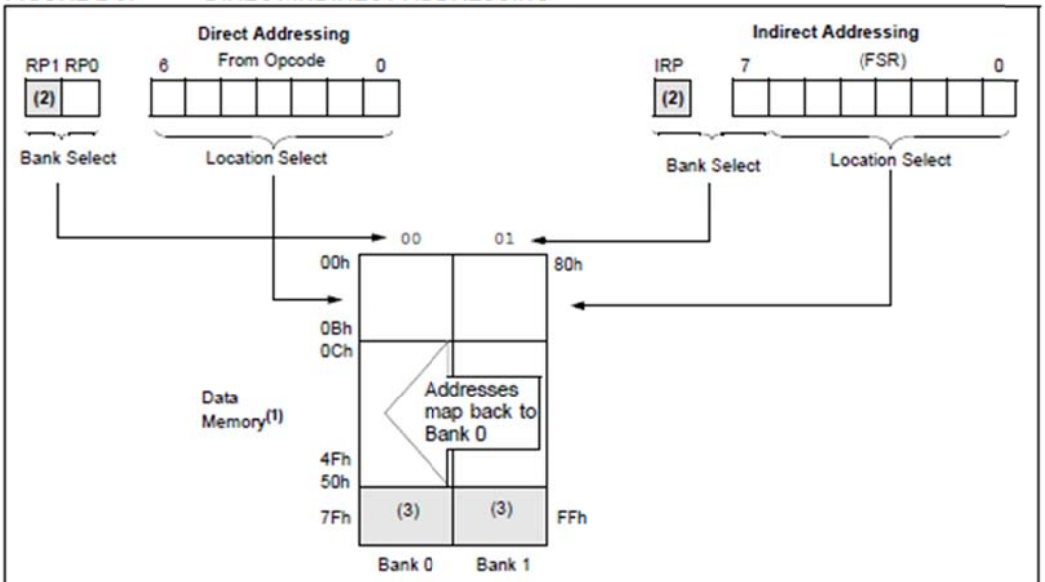
EXAMPLE 2-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```
        movlw   0x20    ;initialize pointer
        movwf   FSR     ;to RAM
NEXT    clrf    INDF    ;clear INDF register
        incf    FSR     ;inc pointer
        btfss   FSR,4   ;all done?
        goto    NEXT    ;NO, clear next
CONTINUE
         :              ;YES, continue
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-3. However, IRP is not used in the PIC16F84A.

PIC16F84A

FIGURE 2-3: DIRECT/INDIRECT ADDRESSING



Note 1: For memory map detail, see Figure 2-2.
2: Maintain as clear for upward compatibility with future products.
3: Not implemented.

3.0 DATA EEPROM MEMORY

The EEPROM data memory is readable and writable during normal operation (full VDD range). This memory is not directly mapped in the register file space. Instead it is indirectly addressed through the Special Function Registers. There are four SFRs used to read and write this memory. These registers are:

- EECON1
- EECON2 (not a physically implemented register)
- EEDATA
- EEADR

EEDATA holds the 8-bit data for read/write, and EEADR holds the address of the EEPROM location being accessed. PIC16F84A devices have 64 bytes of data EEPROM with an address range from 0h to 3Fh.

The EEPROM data memory allows byte read and write. A byte write automatically erases the location and writes the new data (erase before write). The EEPROM data memory is rated for high erase/write cycles. The write time is controlled by an on-chip timer. The write-time will vary with voltage and temperature as well as from chip to chip. Please refer to AC specifications for exact limits.

When the device is code protected, the CPU may continue to read and write the data EEPROM memory. The device programmer can no longer access this memory.

Additional information on the Data EEPROM is available in the PICmicro™ Mid-Range Reference Manual (DS33023).

REGISTER 3-1: EECON1 REGISTER (ADDRESS 88h)

U-0	U-0	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
—	—	—	EEIF	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **WRERR:** EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ Reset or any WDT Reset during normal operation)
0 = The write operation completed

bit 2 **WREN:** EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a write cycle. The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
0 = Does not initiate an EEPROM read

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC16F84A

3.1 Reading the EEPROM Data Memory

To read a data memory location, the user must write the address to the EEADR register and then set control bit RD (EECON1<0>). The data is available, in the very next cycle, in the EEDATA register; therefore, it can be read in the next instruction. EEDATA will hold this value until another read or until it is written to by the user (during a write operation).

EXAMPLE 3-1: DATA EEPROM READ

```
BCF     STATUS, RP0   ; Bank 0
MOVLW   CONFIG_ADDR   ;
MOVWF   EEADR         ; Address to read
BSF     STATUS, RP0   ; Bank 1
BSF     EECON1, RD    ; EE Read
BCF     STATUS, RP0   ; Bank 0
MOVF    EEDATA, W     ; W = EEDATA
```

3.2 Writing to the EEPROM Data Memory

To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific sequence to initiate the write for each byte.

EXAMPLE 3-2: DATA EEPROM WRITE

```
        BSF     STATUS, RP0   ; Bank 1
        BCF     INTCON, GIE   ; Disable INTs.
        BSF     EECON1, WREN  ; Enable Write
        MOVLW   55h           ;
Required
Sequence
        MOVWF   EECON2        ; Write 55h
        MOVLW   AAh           ;
        MOVWF   EECON2        ; Write AAh
        BSF     EECON1,WR     ; Set WR bit
                              ;   begin write
        BSF     INTCON, GIE   ; Enable INTs.
```

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. We strongly recommend that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM due to errant (unexpected) code execution (i.e., lost programs). The user should keep the WREN bit clear at all times, except when updating EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set unless the WREN bit is set.

At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. EEIF must be cleared by software.

3.3 Write Verify

Depending on the application, good programming practice may dictate that the value written to the Data EEPROM should be verified (Example 3-3) to the desired value to be written. This should be used in applications where an EEPROM bit will be stressed near the specification limit.

Generally, the EEPROM write failure will be a bit which was written as a '0', but reads back as a '1' (due to leakage off the bit).

EXAMPLE 3-3: WRITE VERIFY

```
        BCF   STATUS,RP0 ; Bank 0
        :                ; Any code
        :                ; can go here
        MOVF  EEDATA,W   ; Must be in Bank 0
        BSF   STATUS,RP0 ; Bank 1
READ
        BSF   EECON1, RD ; YES, Read the
                         ; value written
        BCF   STATUS, RP0 ; Bank 0
                         ;
                         ; Is the value written
                         ; (in W reg) and
                         ; read (in EEDATA)
                         ; the same?
                         ;
        SUBWF EEDATA, W  ;
        BTFSS STATUS, Z  ; Is difference 0?
        GOTO  WRITE_ERR  ; NO, Write error
```

TABLE 3-1: REGISTERS/BITS ASSOCIATED WITH DATA EEPROM

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
08h	EEDATA	EEPROM Data Register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM Address Register								xxxx xxxx	uuuu uuuu
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM Control Register 2								---- ----	---- ----

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends upon condition.
Shaded cells are not used by data EEPROM.

4.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual (DS33023).

4.1 PORTA and TRISA Registers

PORTA is a 5-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Note:	On a Power-on Reset, these pins are configured as inputs and read as '0'.

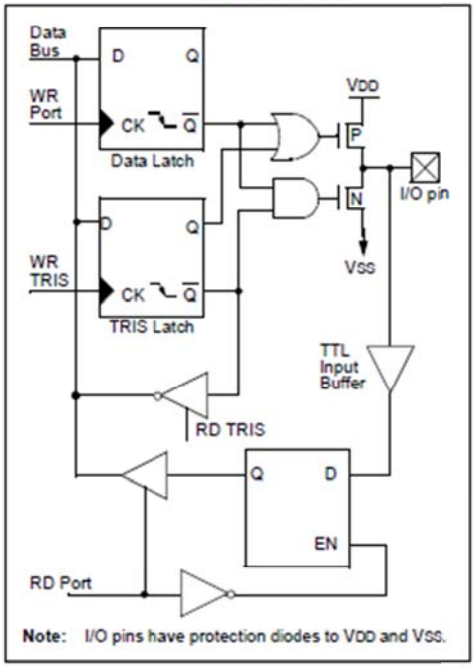
Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read. This value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

EXAMPLE 4-1: INITIALIZING PORTA

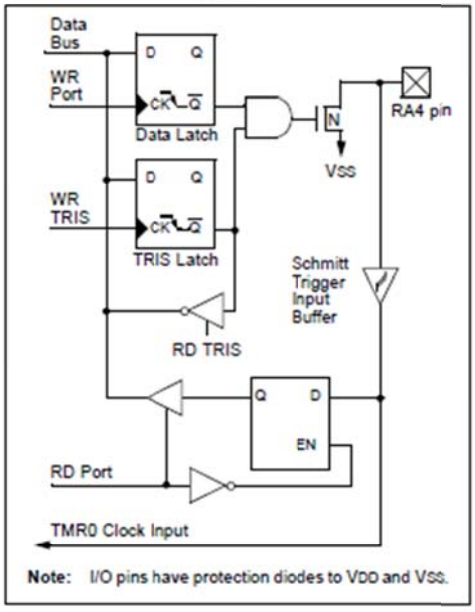
```
BCF     STATUS, RP0 ;
CLRF    PORTA       ; Initialize PORTA by
                    ; clearing output
                    ; data latches
BSF     STATUS, RP0 ; Select Bank 1
MOVLW   0x0F        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISA       ; Set RA<3:0> as inputs
                    ; RA4 as output
                    ; TRISA<7:5> are always
                    ; read as '0'.
```

FIGURE 4-1: BLOCK DIAGRAM OF PINS RA3:RA0



Note: I/O pins have protection diodes to VDD and VSS.

FIGURE 4-2: BLOCK DIAGRAM OF PIN RA4



Note: I/O pins have protection diodes to VDD and VSS.

PIC16F84A

TABLE 4-1: PORTA FUNCTIONS

Name	Bit0	Buffer Type	Function
RA0	bit0	TTL	Input/output
RA1	bit1	TTL	Input/output
RA2	bit2	TTL	Input/output
RA3	bit3	TTL	Input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0. Output is open drain type.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 4-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

4.2 PORTB and TRISB Registers

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

EXAMPLE 4-2: INITIALIZING PORTB

```
BCF     STATUS, RP0 ;
CLRF    PORTB       ; Initialize PORTB by
                    ; clearing output
                    ; data latches
BSF     STATUS, RP0 ; Select Bank 1
MOVLW   0xCF        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISB       ; Set RB<3:0> as inputs
                    ; RB<5:4> as outputs
                    ; RB<7:6> as inputs
```

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (OPTION<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Four of PORTB's pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

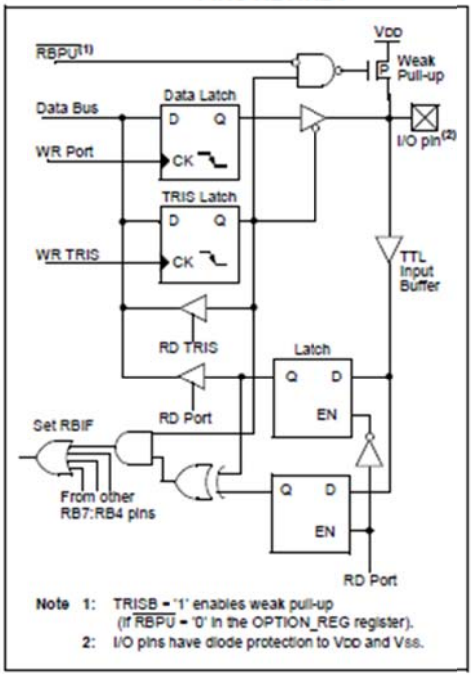
a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

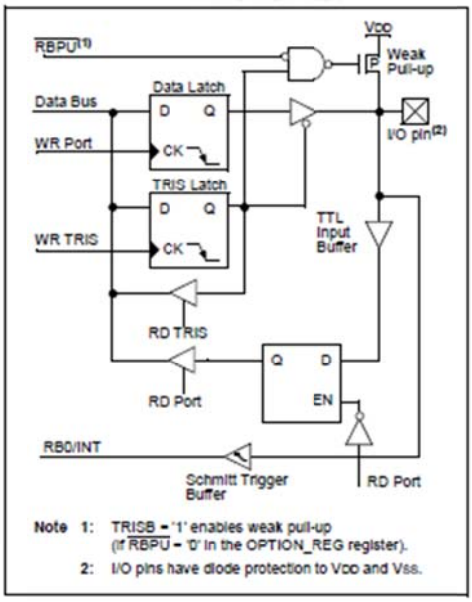
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

FIGURE 4-3: BLOCK DIAGRAM OF PINS RB7:RB4



Note 1: TRISB = '1' enables weak pull-up (if $\overline{\text{RBPU}}$ = '0' in the OPTION_REG register).
2: I/O pins have diode protection to VDD and VSS.

FIGURE 4-4: BLOCK DIAGRAM OF PINS RB3:RB0



Note 1: TRISB = '1' enables weak pull-up (if $\overline{\text{RBPU}}$ = '0' in the OPTION_REG register).
2: I/O pins have diode protection to VDD and VSS.

PIC16F84A

TABLE 4-3: PORTB FUNCTIONS

Name	Bit	Buffer Type	I/O Consistency Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger.
Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

TABLE 4-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

5.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- Internal or external clock select
- Edge select for external clock
- 8-bit software programmable prescaler
- Interrupt-on-overflow from FFh to 00h

Figure 5-1 is a simplified block diagram of the Timer0 module.

Additional information on timer modules is available in the PICmicro™ Mid-Range Reference Manual (DS33023).

5.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In Counter mode, Timer0 will increment, either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed below.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

Additional information on external clock requirements is available in the PICmicro™ Mid-Range Reference Manual, (DS33023).

5.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer, respectively (Figure 5-2). For simplicity, this counter is being referred to as "prescaler" throughout this data sheet. Note that there is only one prescaler available which is mutually exclusively shared between the Timer0 module and the Watchdog Timer. Thus, a prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer, and vice-versa.

The prescaler is not readable or writable.

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

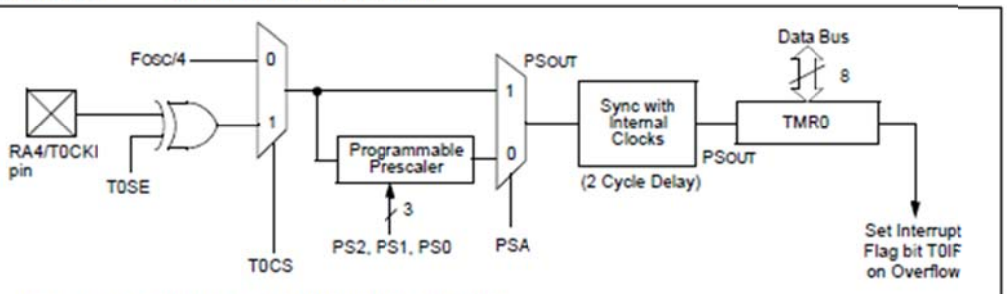
Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable.

Setting bit PSA will assign the prescaler to the Watchdog Timer (WDT). When the prescaler is assigned to the WDT, prescale values of 1:1, 1:2, ..., 1:128 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF 1`, `MOVWF 1`, `BSF 1`, etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the WDT.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

FIGURE 5-1: TIMER0 BLOCK DIAGRAM



Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 5-2 for detailed block diagram).

PIC16F84A

5.2.1 SWITCHING PRESCALER ASSIGNMENT

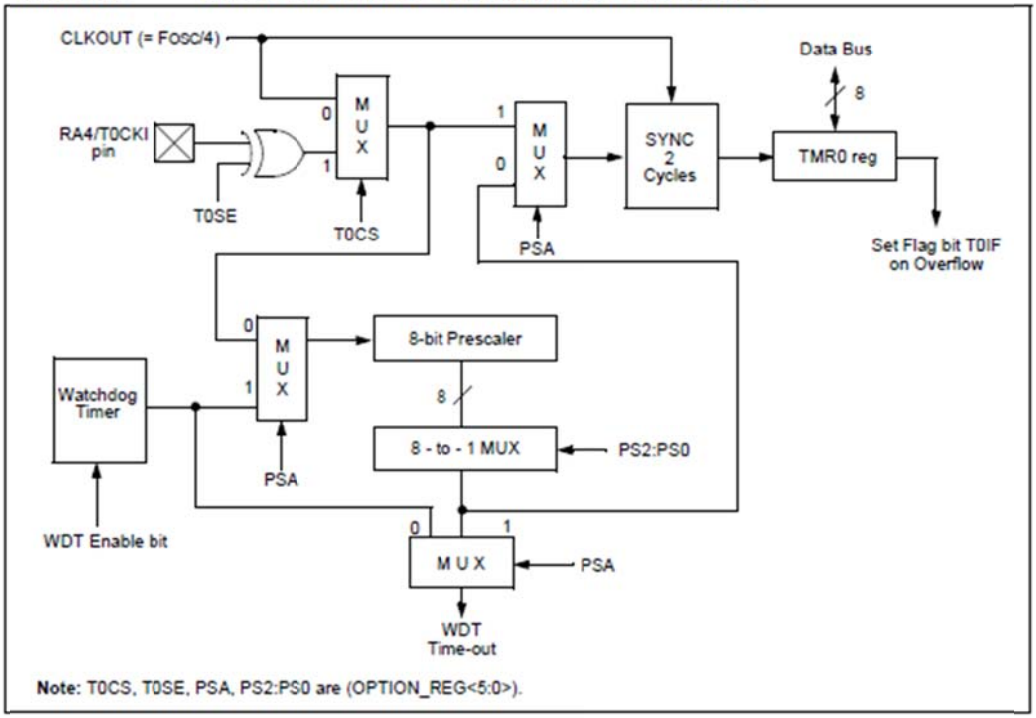
The prescaler assignment is fully under software control (i.e., it can be changed "on the fly" during program execution).

Note: To avoid an unintended device RESET, a specific instruction sequence (shown in the PICmicro™ Mid-Range Reference Manual, DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

5.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP since the timer is shut-off during SLEEP.

FIGURE 5-2: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
01h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

6.0 SPECIAL FEATURES OF THE CPU

What sets a microcontroller apart from other processors are special circuits to deal with the needs of real time applications. The PIC16F84A has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These features are:

- OSC Selection
- RESET
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code Protection
- ID Locations
- In-Circuit Serial Programming™ (ICSP™)

The PIC16F84A has a Watchdog Timer which can be shut-off only through configuration bits. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 72 ms (nominal) on power-up only. This design keeps the device in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode offers a very low current power-down mode. The user can wake-up from SLEEP through external RESET, Watchdog Timer Time-out or through an interrupt. Several oscillator options are provided to allow the part to fit the application. The RC oscillator option saves system cost while the LP crystal option saves power. A set of configuration bits are used to select the various options.

Additional information on special features is available in the PICmicro™ Mid-Range Reference Manual (DS33023).

6.1 Configuration Bits

The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped in program memory location 2007h.

Address 2007h is beyond the user program memory space and it belongs to the special test/configuration memory space (2000h - 3FFFh). This space can only be accessed during programming.

REGISTER 6-1: PIC16F84A CONFIGURATION WORD

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	F0SC1	F0SC0
bit13													bit0

bit 13-4 **CP:** Code Protection bit
1 = Code protection disabled
0 = All program memory is code protected

bit 3 **$\overline{\text{PWRTE}}$:** Power-up Timer Enable bit
1 = Power-up Timer is disabled
0 = Power-up Timer is enabled

bit 2 **WDTE:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0:** Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

PIC16F84A

6.2 Oscillator Configurations

6.2.1 OSCILLATOR TYPES

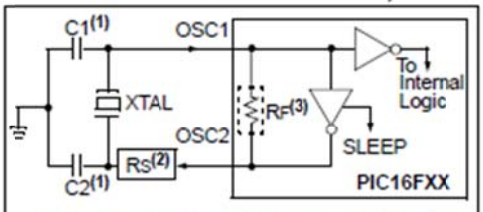
The PIC16F84A can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

- LP Low Power Crystal
- XT Crystal/Resonator
- HS High Speed Crystal/Resonator
- RC Resistor/Capacitor

6.2.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In XT, LP, or HS modes, a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation (Figure 6-1).

FIGURE 6-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



Note 1: See Table 6-1 for recommended values of C1 and C2.

2: A series resistor (RS) may be required for AT strip cut crystals.

The PIC16F84A oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications. When in XT, LP, or HS modes, the device can have an external clock source to drive the OSC1/CLKIN pin (Figure 6-2).

FIGURE 6-2: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)

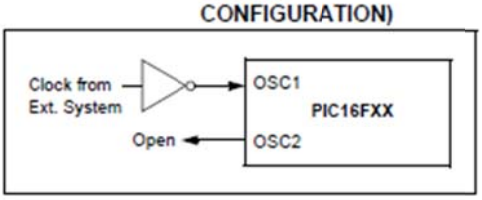


TABLE 6-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Ranges Tested:			
Mode	**Freq**	**OSC1/C1**	**OSC2/C2**
XT	455 kHz	47 - 100 pF	47 - 100 pF
	2.0 MHz	15 - 33 pF	15 - 33 pF
	4.0 MHz	15 - 33 pF	15 - 33 pF
HS	8.0 MHz	15 - 33 pF	15 - 33 pF
	10.0 MHz	15 - 33 pF	15 - 33 pF

Note: Recommended values of C1 and C2 are identical to the ranges tested in this table. Higher capacitance increases the stability of the oscillator, but also increases the start-up time. These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for the appropriate values of external components.

Note: When using resonators with frequencies above 3.5 MHz, the use of HS mode rather than XT mode, is recommended. HS mode may be used at any VDD for which the controller is rated.

TABLE 6-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

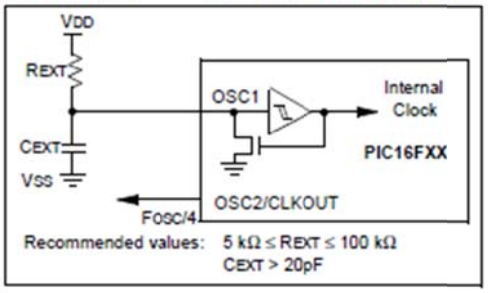
Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz	68 - 100 pF	68 - 100 pF
	200 kHz	15 - 33 pF	15 - 33 pF
XT	100 kHz	100 - 150 pF	100 - 150 pF
	2 MHz	15 - 33 pF	15 - 33 pF
	4 MHz	15 - 33 pF	15 - 33 pF
HS	4 MHz	15 - 33 pF	15 - 33 pF
	20 MHz	15 - 33 pF	15 - 33 pF

Note: Higher capacitance increases the stability of the oscillator, but also increases the start-up time. These values are for design guidance only. Rs may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.
For VDD > 4.5V, C1 = C2 ≈ 30 pF is recommended.

6.2.3 RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) values, capacitor (CEXT) values, and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types also affects the oscillation frequency, especially for low CEXT values. The user needs to take into account variation, due to tolerance of the external R and C components. Figure 6-3 shows how an R/C combination is connected to the PIC16F84A.

FIGURE 6-3: RC OSCILLATOR MODE



PIC16F84A

6.3 RESET

The PIC16F84A differentiates between various kinds of RESET:

- Power-on Reset (POR)
- $\overline{MCLR}$ during normal operation
- $\overline{MCLR}$ during SLEEP
- WDT Reset (during normal operation)
- WDT Wake-up (during SLEEP)

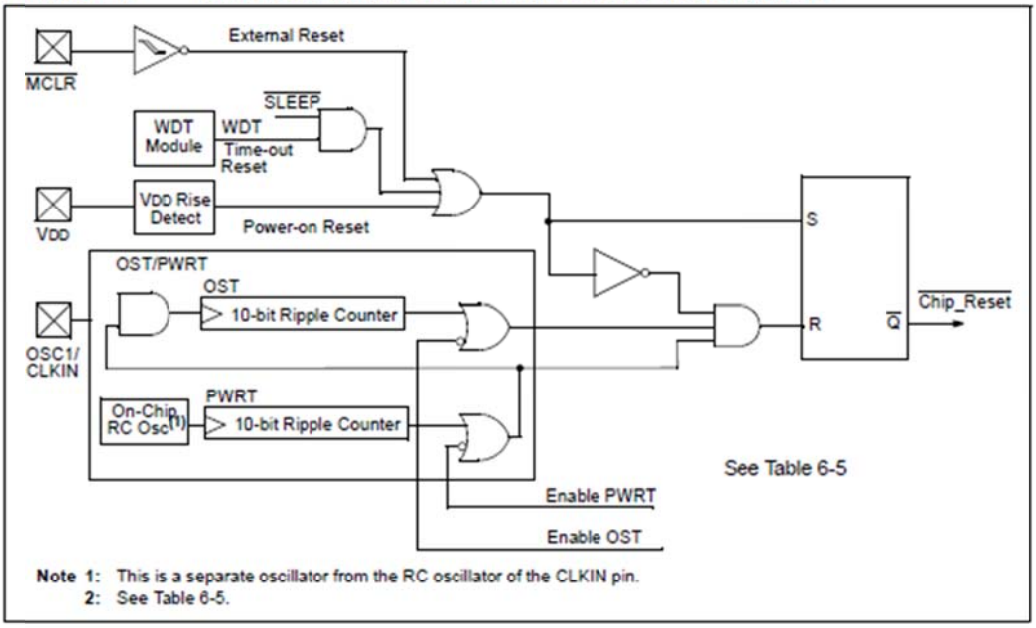
Figure 6-4 shows a simplified block diagram of the On-Chip RESET Circuit. The $\overline{MCLR}$ Reset path has a noise filter to ignore small pulses. The electrical specifications state the pulse width requirements for the $\overline{MCLR}$ pin.

Some registers are not affected in any RESET condition; their status is unknown on a POR and unchanged in any other RESET. Most other registers are reset to a "RESET state" on POR, $\overline{MCLR}$ or WDT Reset during normal operation and on $\overline{MCLR}$ during SLEEP. They are not affected by a WDT Reset during SLEEP, since this RESET is viewed as the resumption of normal operation.

Table 6-3 gives a description of RESET conditions for the program counter (PC) and the STATUS register. Table 6-4 gives a full description of RESET states for all registers.

The $\overline{TO}$ and $\overline{PD}$ bits are set or cleared differently in different RESET situations (Section 6.7). These bits are used in software to determine the nature of the RESET.

FIGURE 6-4: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.
2: See Table 6-5.

TABLE 6-3: RESET CONDITION FOR PROGRAM COUNTER AND THE STATUS REGISTER

Condition	Program Counter	STATUS Register
Power-on Reset	000h	0001 1xxx
$\overline{MCLR}$ during normal operation	000h	000u uuuu
$\overline{MCLR}$ during SLEEP	000h	0001 0uuu
WDT Reset (during normal operation)	000h	0000 1uuu
WDT Wake-up	PC + 1	uuu0 0uuu
Interrupt wake-up from SLEEP	PC + 1(1)	uuu1 0uuu

Legend: u = unchanged, x = unknown

Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

TABLE 6-4: RESET CONDITIONS FOR ALL REGISTERS

Register	Address	Power-on Reset	$\overline{\text{MCLR}}$ during: – normal operation – SLEEP WDT Reset during normal operation	Wake-up from SLEEP: – through interrupt – through WDT Time-out
W	—	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	---- ----	---- ----	---- ----
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000 0000	0000 0000	PC + 1(2)
STATUS	03h	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA(4)	05h	---x xxxx	---u uuuu	---u uuuu
PORTB(5)	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADR	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000u	uuuu uuuu(1)
INDF	80h	---- ----	---- ----	---- ----
OPTION_REG	81h	1111 1111	1111 1111	uuuu uuuu
PCL	82h	0000 0000	0000 0000	PC + 1(2)
STATUS	83h	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	84h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TRISA	85h	---1 1111	---1 1111	---u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
EECON1	88h	---0 x000	---0 q000	---0 uuuu
EECON2	89h	---- ----	---- ----	---- ----
PCLATH	8Ah	---0 0000	---0 0000	---u uuuu
INTCON	8Bh	0000 000x	0000 000u	uuuu uuuu(1)

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition

Note 1: One or more bits in INTCON will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

3: Table 6-3 lists the RESET value for each specific condition.

4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

PIC16F84A

6.4 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.2V - 1.7V). To take advantage of the POR, just tie the MCLR pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-on Reset. A minimum rise time for VDD must be met for this to operate properly. See Electrical Specifications for details.

When the device starts normal operation (exits the RESET condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in RESET until the operating conditions are met.

For additional information, refer to Application Note AN607, "*Power-up Trouble Shooting*."

The POR circuit does not produce an internal RESET when VDD declines.

6.5 Power-up Timer (PWRT)

The Power-up Timer (PWRT) provides a fixed 72 ms nominal time-out (TPWRT) from POR (Figures 6-6 through 6-9). The Power-up Timer operates on an internal RC oscillator. The chip is kept in RESET as long as the PWRT is active. The PWRT delay allows the VDD to rise to an acceptable level (possible exception shown in Figure 6-9).

A configuration bit, PWRTE, can enable/disable the PWRT. See Register 6-1 for the operation of the PWRTE bit for a particular device.

The power-up time delay TPWRT will vary from chip to chip due to VDD, temperature, and process variation. See DC parameters for details.

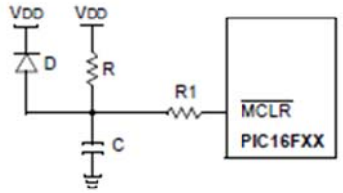
6.6 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle delay (from OSC1 input) after the PWRT delay ends (Figure 6-6, Figure 6-7, Figure 6-8 and Figure 6-9). This ensures the crystal oscillator or resonator has started and stabilized.

The OST time-out (TOST) is invoked only for XT, LP and HS modes and only on Power-on Reset or wake-up from SLEEP.

When VDD rises very slowly, it is possible that the TPWRT time-out and TOST time-out will expire before VDD has reached its final value. In this case (Figure 6-9), an external Power-on Reset circuit may be necessary (Figure 6-5).

FIGURE 6-5: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Note 1: External Power-on Reset circuit is required only if VDD power-up rate is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that voltage drop across R does not exceed 0.2V (max leakage current spec on MCLR pin is 5 µA). A larger voltage drop will degrade VIH level on the MCLR pin.

3: R1 = 100Ω to 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of a MCLR pin breakdown due to ESD or EOS.

FIGURE 6-6: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

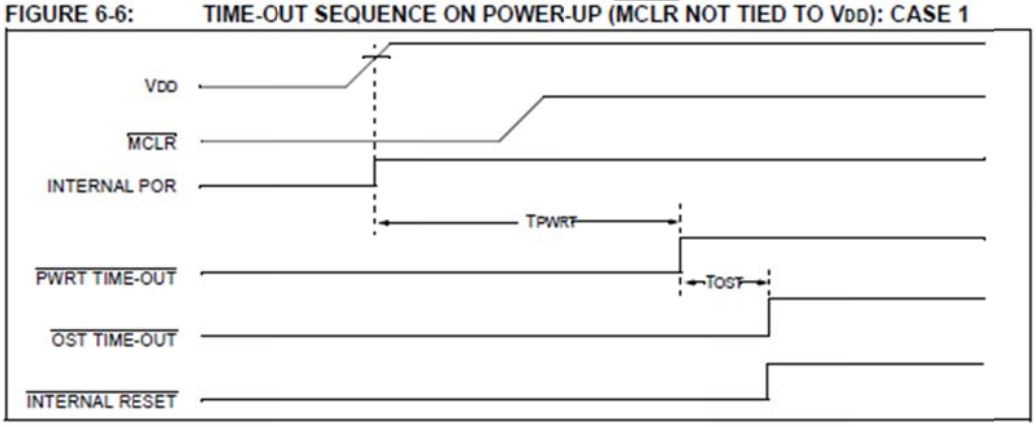


FIGURE 6-7: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

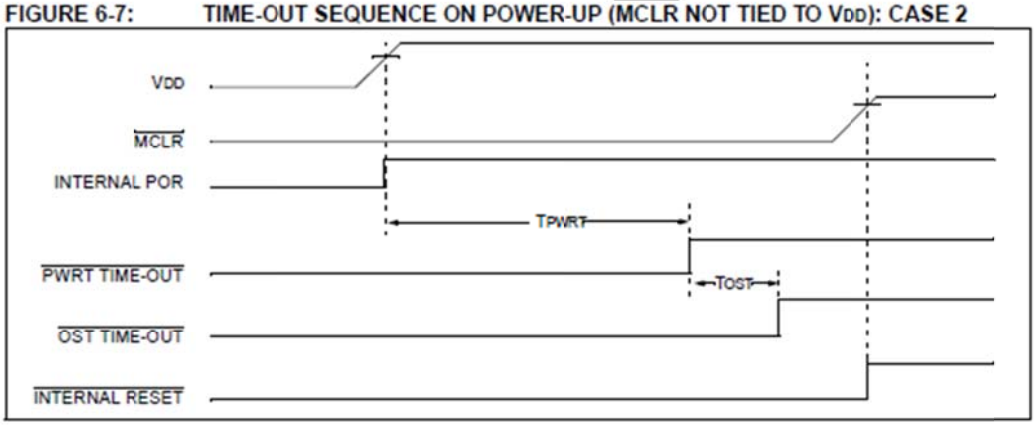


FIGURE 6-8: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD): FAST VDD RISE TIME

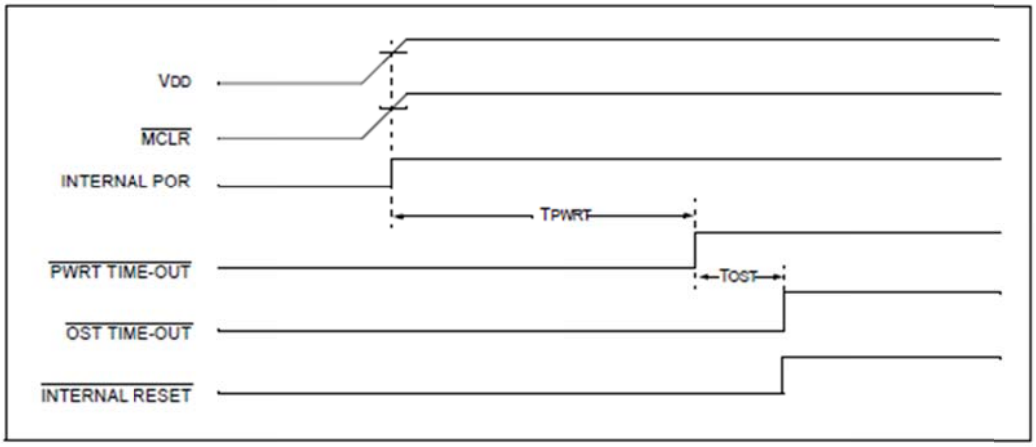
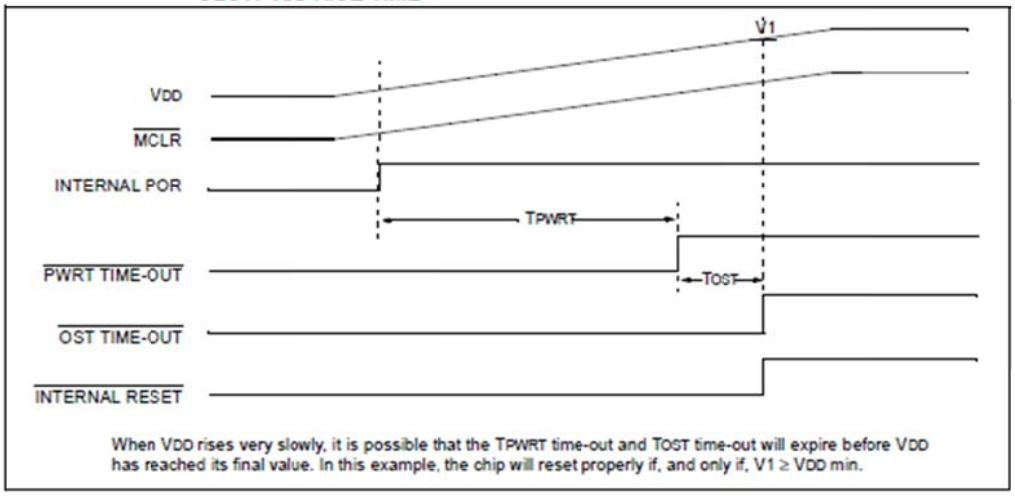


FIGURE 6-9: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO VDD): SLOW VDD RISE TIME



When VDD rises very slowly, it is possible that the TPWRT time-out and TOST time-out will expire before VDD has reached its final value. In this example, the chip will reset properly if, and only if, V1 ≥ VDD min.

6.7 Time-out Sequence and Power-down Status Bits ($\overline{TO}$/$\overline{PD}$)

On power-up (Figures 6-6 through 6-9), the time-out sequence is as follows:

1. PWRT time-out is invoked after a POR has expired.
2. Then, the OST is activated.

The total time-out will vary based on oscillator configuration and PWRTE configuration bit status. For example, in RC mode with the PWRT disabled, there will be no time-out at all.

TABLE 6-5: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up		Wake-up from SLEEP
	PWRT Enabled	PWRT Disabled	
XT, HS, LP	72 ms + 1024TOSC	1024TOSC	1024TOSC
RC	72 ms	—	—

Since the time-outs occur from the POR pulse, if $\overline{MCLR}$ is kept low long enough, the time-outs will expire. Then bringing $\overline{MCLR}$ high, execution will begin immediately (Figure 6-6). This is useful for testing purposes or to synchronize more than one PIC16F84A device when operating in parallel.

Table 6-6 shows the significance of the $\overline{TO}$ and $\overline{PD}$ bits. Table 6-3 lists the RESET conditions for some special registers, while Table 6-4 lists the RESET conditions for all the registers.

TABLE 6-6: STATUS BITS AND THEIR SIGNIFICANCE

$\overline{TO}$	$\overline{PD}$	Condition
1	1	Power-on Reset
0	x	Illegal, $\overline{TO}$ is set on $\overline{POR}$
x	0	Illegal, $\overline{PD}$ is set on $\overline{POR}$
0	1	WDT Reset (during normal operation)
0	0	WDT Wake-up
1	1	$\overline{MCLR}$ during normal operation
1	0	$\overline{MCLR}$ during SLEEP or interrupt wake-up from SLEEP

6.8 Interrupts

The PIC16F84A has 4 sources of interrupt:

- External interrupt RB0/INT pin
- TMR0 overflow interrupt
- PORTB change interrupts (pins RB7:RB4)
- Data EEPROM write complete interrupt

The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also contains the individual and global interrupt enable bits.

The global interrupt enable bit, GIE (INTCON<7>), enables (if set) all unmasked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in INTCON register. Bit GIE is cleared on RESET.

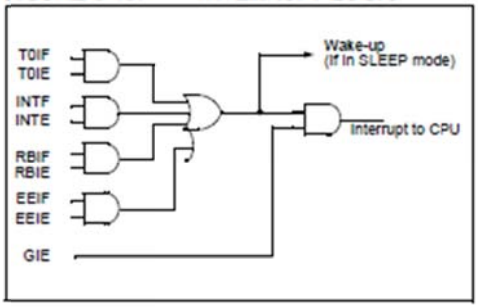
The "return from interrupt" instruction, RETFIE, exits interrupt routine as well as sets the GIE bit, which re-enables interrupts.

The RB0/INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

When an interrupt is responded to, the GIE bit is cleared to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with 0004h. For external interrupt events, such as the RB0/INT pin or PORTB change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency depends when the interrupt event occurs. The latency is the same for both one and two cycle instructions. Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid infinite interrupt requests.

Note:	Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

FIGURE 6-10: INTERRUPT LOGIC



6.8.1 INT INTERRUPT

External interrupt on RB0/INT pin is edge triggered: either rising if INTEDG bit (OPTION_REG<6>) is set, or falling if INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, the INTF bit (INTCON<1>) is set. This interrupt can be disabled by clearing control bit INTE (INTCON<4>). Flag bit INTF must be cleared in software via the Interrupt Service Routine before re-enabling this interrupt. The INT interrupt can wake the processor from SLEEP (Section 6.11) only if the INTE bit was set prior to going into SLEEP. The status of the GIE bit decides whether the processor branches to the interrupt vector following wake-up.

6.8.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in TMR0 will set flag bit T0IF (INTCON<2>). The interrupt can be enabled/disabled by setting/clearing enable bit T0IE (INTCON<5>) (Section 5.0).

6.8.3 PORTB INTERRUPT

An input change on PORTB<7:4> sets flag bit RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON<3>) (Section 4.2).

Note:	For a change on the I/O pin to be recognized, the pulse width must be at least TCY wide.

6.8.4 DATA EEPROM INTERRUPT

At the completion of a data EEPROM write cycle, flag bit EEIF (EECON1<4>) will be set. The interrupt can be enabled/disabled by setting/clearing enable bit EEIE (INTCON<6>) (Section 3.0).

PIC16F84A

6.9 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users wish to save key register values during an interrupt (e.g., W register and STATUS register). This is implemented in software.

The code in Example 6-1 stores and restores the STATUS and W register's values. The user defined registers, W_TEMP and STATUS_TEMP are the temporary storage locations for the W and STATUS registers values.

Example 6-1 does the following:

a) Stores the W register.
b) Stores the STATUS register in STATUS_TEMP.
c) Executes the Interrupt Service Routine code.
d) Restores the STATUS (and bank select bit) register.
e) Restores the W register.

EXAMPLE 6-1: SAVING STATUS AND W REGISTERS IN RAM

```
PUSH    MOVWF   W_TEMP              ; Copy W to TEMP register,
        SWAPF   STATUS,     W       ; Swap status to be saved into W
        MOVWF   STATUS_TEMP         ; Save status to STATUS_TEMP register
ISR     :                           :
        :                           ; Interrupt Service Routine
        :                           ; should configure Bank as required
        :                           ;
POP     SWAPF   STATUS_TEMP,W       ; Swap nibbles in STATUS_TEMP register
                                    ; and place result into W
        MOVWF   STATUS              ; Move W into STATUS register
                                    ; (sets bank to original state)
        SWAPF   W_TEMP,     F       ; Swap nibbles in W_TEMP and place result in W_TEMP
        SWAPF   W_TEMP,     W       ; Swap nibbles in W_TEMP and place result into W
```

6.10 Watchdog Timer (WDT)

The Watchdog Timer is a free running On-Chip RC Oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT time-out generates a device RESET. If the device is in SLEEP mode, a WDT wake-up causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming configuration bit WDTE as a '0' (Section 6.1).

6.10.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms, (with no prescaler). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION_REG register. Thus, time-out periods up to 2.3 seconds can be realized.

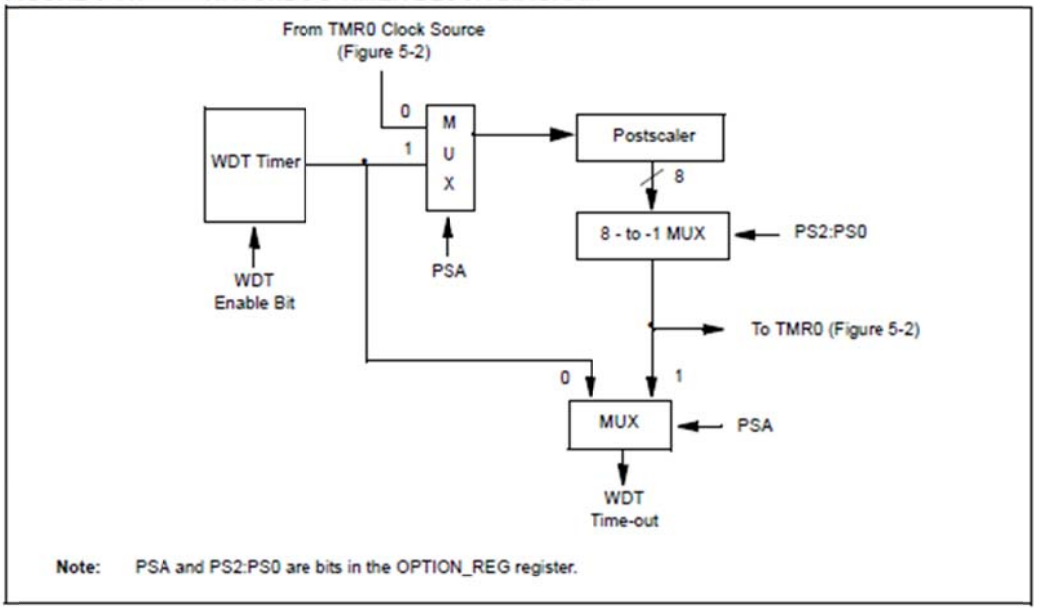
The CLRWDT and SLEEP instructions clear the WDT and the postscaler (if assigned to the WDT) and prevent it from timing out and generating a device RESET condition.

The $\overline{TO}$ bit in the STATUS register will be cleared upon a WDT time-out.

6.10.2 WDT PROGRAMMING CONSIDERATIONS

It should also be taken into account that under worst case conditions (V_{DD} = Min., Temperature = Max., Max. WDT Prescaler), it may take several seconds before a WDT time-out occurs.

FIGURE 6-11: WATCHDOG TIMER BLOCK DIAGRAM



Note: PSA and PS2:PS0 are bits in the OPTION_REG register.

TABLE 6-7: SUMMARY OF REGISTERS ASSOCIATED WITH THE WATCHDOG TIMER

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
2007h	Config. bits	(2)	(2)	(2)	(2)	$\overline{PWRTE}$(1)	WDTE	FOSC1	FOSC0	(2)	
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown. Shaded cells are not used by the WDT.

Note 1: See Register 6-1 for operation of the $\overline{PWRTE}$ bit.

2: See Register 6-1 and Section 6.12 for operation of the code and data protection bits.

6.11 Power-down Mode (SLEEP)

A device may be powered down (SLEEP) and later powered up (wake-up from SLEEP).

6.11.1 SLEEP

The Power-down mode is entered by executing the SLEEP instruction.

If enabled, the Watchdog Timer is cleared (but keeps running), the $\overline{PD}$ bit (STATUS<3>) is cleared, the $\overline{TO}$ bit (STATUS<4>) is set, and the oscillator driver is turned off. The I/O ports maintain the status they had before the SLEEP instruction was executed (driving high, low, or hi-impedance).

For the lowest current consumption in SLEEP mode, place all I/O pins at either VDD or VSS, with no external circuitry drawing current from the I/O pins, and disable external clocks. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should also be at VDD or VSS. The contribution from on-chip pull-ups on PORTB should be considered.

The $\overline{MCLR}$ pin must be at a logic high level (VIHMC).

It should be noted that a RESET generated by a WDT time-out does not drive the $\overline{MCLR}$ pin low.

6.11.2 WAKE-UP FROM SLEEP

The device can wake-up from SLEEP through one of the following events:

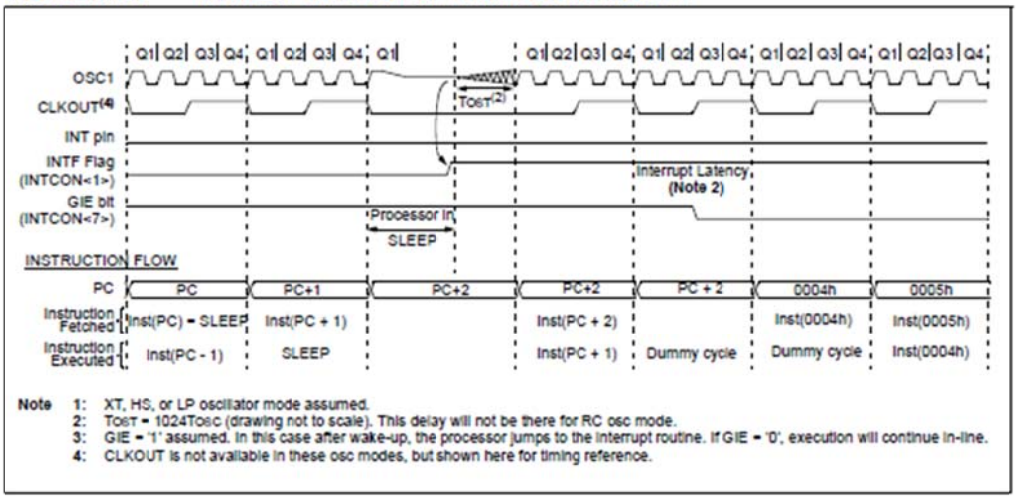
1. External RESET input on $\overline{MCLR}$ pin.
2. WDT wake-up (if WDT was enabled).
3. Interrupt from RB0/INT pin, RB port change, or data EEPROM write complete.

Peripherals cannot generate interrupts during SLEEP, since no on-chip Q clocks are present.

The first event ($\overline{MCLR}$ Reset) will cause a device RESET. The two latter events are considered a continuation of program execution. The $\overline{TO}$ and $\overline{PD}$ bits can be used to determine the cause of a device RESET. The $\overline{PD}$ bit, which is set on power-up, is cleared when SLEEP is invoked. The $\overline{TO}$ bit is cleared if a WDT time-out occurred (and caused wake-up).

While the SLEEP instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up occurs regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

FIGURE 6-12: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: XT, HS, or LP oscillator mode assumed.
2: TOST = 1024TOSC (drawing not to scale). This delay will not be there for RC osc mode.
3: GIE = '1' assumed. In this case after wake-up, the processor jumps to the interrupt routine. If GIE = '0', execution will continue in-line.
4: CLKOUT is not available in these osc modes, but shown here for timing reference.

6.11.3 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a SLEEP instruction, the SLEEP instruction will complete as a NOP. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bits will not be cleared.
- If the interrupt occurs **during or after** the execution of a SLEEP instruction, the device will immediately wake-up from SLEEP. The SLEEP instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit will be cleared.

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the SLEEP instruction was executed as a NOP.

To ensure that the WDT is cleared, a CLRWDT instruction should be executed before a SLEEP instruction.

6.12 Program Verification/Code Protection

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

6.13 ID Locations

Four memory locations (2000h - 2004h) are designated as ID locations to store checksum or other code identification numbers. These locations are not accessible during normal execution but are readable and writable only during program/verify. Only the four Least Significant bits of ID location are usable.

6.14 In-Circuit Serial Programming

PIC16F84A microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground, and the programming voltage. Customers can manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product, allowing the most recent firmware or custom firmware to be programmed.

For complete details of Serial Programming, please refer to the In-Circuit Serial Programming™ (ICSP™) Guide, (DS30277).

7.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-bit word, divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The PIC16CXX instruction set summary in Table 7-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 7-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

TABLE 7-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs.

Table 7-2 lists the instructions recognized by the MPASM™ Assembler.

Figure 7-1 shows the general formats that the instructions can have.

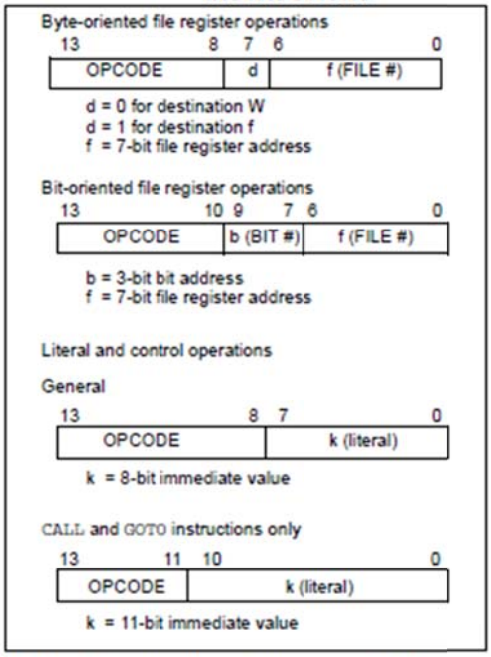
Note: To maintain upward compatibility with future PIC16CXX products, do not use the OPTION and TRIS instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

FIGURE 7-1: GENERAL FORMAT FOR INSTRUCTIONS



A description of each instruction is available in the PICmicro™ Mid-Range Reference Manual (DS33023).

PIC16F84A

TABLE 7-2: PIC16CXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

7.1 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

PIC16F84A

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b' in register 'f' is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead, making it a 2TCY instruction.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Description:	GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

IORLW	Inclusive OR Literal with W
Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the eight-bit literal 'k'. The result is placed in the W register.

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

IORWF	Inclusive OR W with f
Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

PIC16F84A

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Description:	The eight-bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

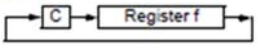
RETURN	Return from Subroutine
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

RLF — **Rotate Left f through Carry**

Syntax:	[*label*] RLF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.

RRF — **Rotate Right f through Carry**

Syntax:	[*label*] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

SUBLW — **Subtract W from Literal**

Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

SUBWF — **Subtract W from f**

Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - (W) → (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

SWAPF — **Swap Nibbles in f**

Syntax:	[*label*] SWAPF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W register. If 'd' is 1, the result is placed in register 'f'.

PIC16F84A

XORLW	Exclusive OR Literal with W
Syntax:	[*label*] XORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register.

XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

8.0 DEVELOPMENT SUPPORT

The PICmicro® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

8.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or 'C')
- One touch assemble (or compile) and download to PICmicro emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

8.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PICmicro MCU's.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

8.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI 'C' compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

8.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can also link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian is a librarian for pre-compiled code to be used with the MPLINK object linker. When a routine from a library is called from another source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications. The MPLIB object librarian manages the creation and modification of library files.

The MPLINK object linker features include:

- Integration with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- Allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- Easier linking because single libraries can be included instead of many smaller files.
- Helps keep code maintainable by grouping related modules together.
- Allows libraries to be created and modules to be added, listed, replaced, deleted or extracted.

8.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user-defined key press, to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and the MPLAB C18 C compilers and the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

8.6 MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB ICE universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE in-circuit emulator system has been designed as a real-time emulation system, with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows® environment were chosen to best make these features available to you, the end user.

8.7 ICEPIC In-Circuit Emulator

The ICEPIC low cost, in-circuit emulator is a solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X and PIC16CXXX families of 8-bit One-Time-Programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules, or daughter boards. The emulator is capable of emulating without target application circuitry being present.

8.8 MPLAB ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the FLASH PICmicro MCUs and can be used to develop for this and other PICmicro microcontrollers. The MPLAB ICD utilizes the in-circuit debugging capability built into the FLASH devices. This feature, along with Microchip's In-Circuit Serial Programming™ protocol, offers cost-effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

8.9 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full-featured programmer, capable of operating in stand-alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable VDD and VPP supplies, which allow it to verify programmed memory at VDD min and VDD max for maximum reliability. It has an LCD display for instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode, the PRO MATE II device programmer can read, verify, or program PICmicro devices. It can also set code protection in this mode.

8.10 PICSTART Plus Entry Level Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

The PICSTART Plus development programmer supports all PICmicro devices with up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

8.11 PICDEM 1 Low Cost PICmicro Demonstration Board

The PICDEM 1 demonstration board is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The user can program the sample microcontrollers provided with the PICDEM 1 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The user can also connect the PICDEM 1 demonstration board to the MPLAB ICE in-circuit emulator and download the firmware to the emulator for testing. A prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs connected to PORTB.

8.12 PICDEM 2 Low Cost PIC16CXX Demonstration Board

The PICDEM 2 demonstration board is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 2 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a serial EEPROM to demonstrate usage of the I2CTM bus and separate headers for connection to an LCD module and a keypad.

8.13 PICDEM 3 Low Cost PIC16CXXX Demonstration Board

The PICDEM 3 demonstration board is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with an LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 3 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer with an adapter socket, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 3 demonstration board to test firmware. A prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM 3 demonstration board is a LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM 3 demonstration board provides an additional RS-232 interface and Windows software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

8.14 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included and the user may erase it and program it with the other sample programs using the PRO MATE II device programmer, or the PICSTART Plus development programmer, and easily debug and test the sample code. In addition, the PICDEM 17 demonstration board supports downloading of programs to and executing out of external FLASH memory on board. The PICDEM 17 demonstration board is also usable with the MPLAB ICE in-circuit emulator, or the PICMASTER emulator and all of the sample programs can be run and modified using either emulator. Additionally, a generous prototype area is available for user hardware.

8.15 KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchip's HCS Secure Data Products. The HCS evaluation kit includes a LCD display to show changing codes, a decoder to decode transmissions and a programming interface to program test transmitters.

TABLE 8-1: DEVELOPMENT TOOLS FROM MICROCHIP

		PIC12CXXX	PIC14000	PIC16C5X	PIC16CEX	PIC16CXXX	PIC16F62X	PIC16C7X	PIC16C7XX	PIC16C8X	PIC16F8XX	PIC16C9XX	PIC17C4X	PIC17C7XX	PIC18CXX2	PIC18FXXX	24CXX/ 25CXX/ 93CXX	HCSXXX	MCRFXXX	MCP2510
Software Tools	MPLAB® Integrated Development Environment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	MPLAB® C17 C Compiler												✓	✓						
	MPLAB® C18 C Compiler														✓	✓				
	MPASM™ Assembler/ MPLINK™ Object Linker	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Emulators	MPLAB® ICE In-Circuit Emulator	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	ICEPIC™ In-Circuit Emulator	✓		✓		✓		✓	✓	✓		✓								
Debugger	MPLAB® ICD In-Circuit Debugger				✓*			✓*			✓					✓				
Programmers	PICSTART® Plus Entry Level Development Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	PRO MATE® II Universal Device Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Demo Boards and Eval Kits	PICDEM™ 1 Demonstration Board			✓		✓		✓†	✓	✓			✓							
	PICDEM™ 2 Demonstration Board				✓†			✓†							✓	✓				
	PICDEM™ 3 Demonstration Board											✓								
	PICDEM™ 14A Demonstration Board		✓																	
	PICDEM™ 17 Demonstration Board													✓						
	KEELOQ® Evaluation Kit																	✓		
	KEELOQ® Transponder Kit																	✓		
	microID™ Programmer's Kit																		✓	
	125 kHz microID™ Developer's Kit																		✓	
	125 kHz Anticollision microID™ Developer's Kit																		✓	
	13.56 MHz Anticollision microID™ Developer's Kit																		✓	
	MCP2510 CAN Developer's Kit																			✓

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.
** Contact Microchip Technology Inc. for availability date.
† Development tool is available on select devices.

9.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias	-55°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on any pin with respect to VSS (except VDD, $\overline{MCLR}$, and RA4)	-0.3V to (VDD + 0.3V)
Voltage on VDD with respect to VSS	-0.3 to +7.5V
Voltage on $\overline{MCLR}$ with respect to VSS(1)	-0.3 to +14V
Voltage on RA4 with respect to VSS	-0.3 to +8.5V
Total power dissipation(2)	800 mW
Maximum current out of VSS pin	150 mA
Maximum current into VDD pin	100 mA
Input clamp current, IIK (VI < 0 or VI > VDD)	± 20 mA
Output clamp current, IOK (VO < 0 or VO > VDD)	± 20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by PORTA	80 mA
Maximum current sourced by PORTA	50 mA
Maximum current sunk by PORTB	150 mA
Maximum current sourced by PORTB	100 mA

Note 1: Voltage spikes below VSS at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to VSS.

2: Power dissipation is calculated as follows: $Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIC16F84A

FIGURE 9-1: PIC16F84A-20 VOLTAGE-FREQUENCY GRAPH

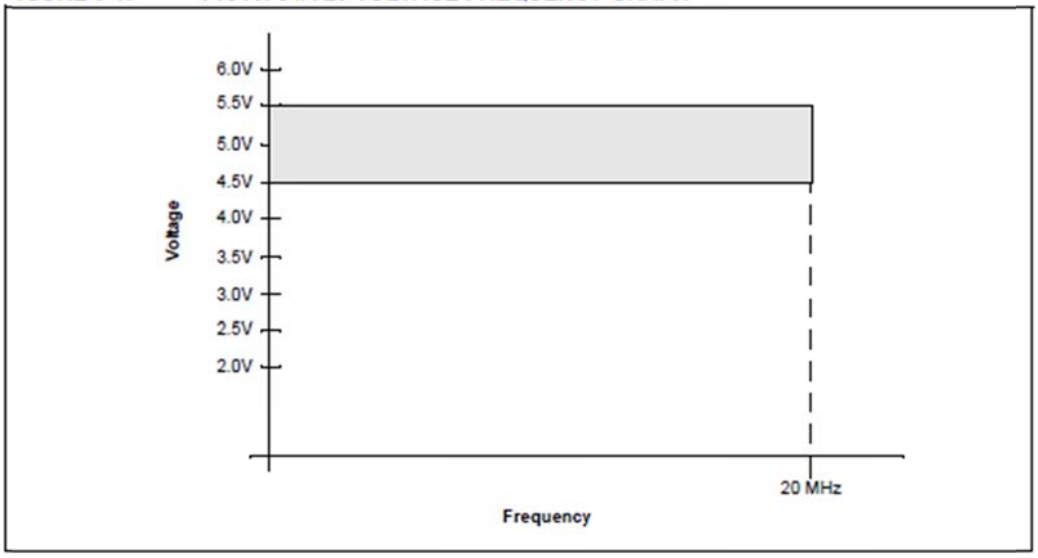
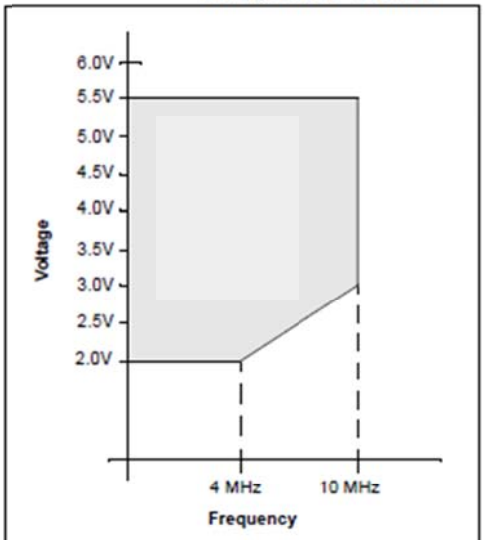


FIGURE 9-2: PIC16LF84A-04 VOLTAGE-FREQUENCY GRAPH

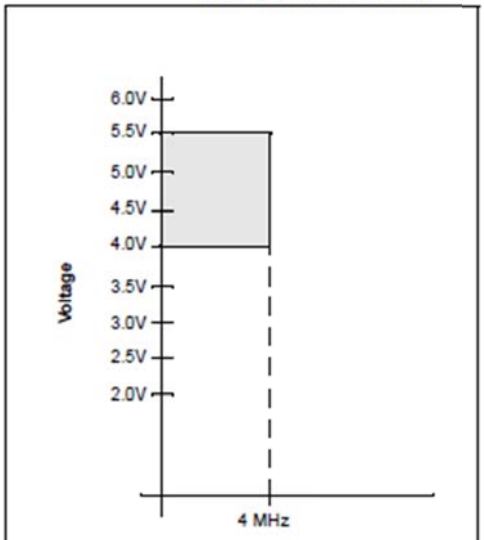


FMAX = (6.0 MHz/V) (VDDAPPMIN - 2.0V) + 4 MHz

Note 1: VDDAPPMIN is the minimum voltage of the PICmicro® device in the application.

2: FMAX has a maximum frequency of 10 MHz.

FIGURE 9-3: PIC16F84A-04 VOLTAGE-FREQUENCY GRAPH



9.1 DC Characteristics

PIC16LF84A-04 (Commercial, Industrial)	Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)
PIC16F84A-04 (Commercial, Industrial, Extended) PIC16F84A-20 (Commercial, Industrial, Extended)	Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)

Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
	VDD	**Supply Voltage**					
D001		16LF84A	2.0	—	5.5	V	XT, RC, and LP osc configuration
D001		16F84A	4.0	—	5.5	V	XT, RC and LP osc configuration
D001A			4.5	—	5.5	V	HS osc configuration
D002	VDR	**RAM Data Retention Voltage (Note 1)**	1.5	—	—	V	Device in SLEEP mode
D003	VPOR	**VDD Start Voltage to ensure internal Power-on Reset signal**	—	Vss	—	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate to ensure internal Power-on Reset signal**	0.05	—	—	V/ms	
	IDD	**Supply Current (Note 2)**					
D010		16LF84A	—	1	4	mA	RC and XT osc configuration **(Note 4)** FOSC = 2.0 MHz, VDD = 5.5V
D010		16F84A	—	1.8	4.5	mA	RC and XT osc configuration **(Note 4)** FOSC = 4.0 MHz, VDD = 5.5V
D010A			—	3	10	mA	RC and XT osc configuration **(Note 4)** FOSC = 4.0 MHz, VDD = 5.5V (During FLASH programming)
D013			—	10	20	mA	HS osc configuration (PIC16F84A-20) FOSC = 20 MHz, VDD = 5.5V
D014		16LF84A	—	15	45	µA	LP osc configuration FOSC = 32 kHz, VDD = 2.0V, WDT disabled

Legend: Rows with standard voltage device data only are shaded for improved readability.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

NR Not rated for operation.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD,
T0CKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD measurement.

PIC16F84A

9.1 DC Characteristics (Continued)

PIC16LF84A-04 (Commercial, Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)				
PIC16F84A-04 (Commercial, Industrial, Extended) **PIC16F84A-20** (Commercial, Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) -40°C ≤ TA ≤ +125°C (extended)				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	IPD	**Power-down Current (Note 3)**					
D020		16LF84A					
D020		16F84A-20 16F84A-04					
D021A		16LF84A	—	0.4	1.0	µA	VDD = 2.0V, WDT disabled, industrial
D021A		16F84A-20	—	1.5	3.5	µA	VDD = 4.5V, WDT disabled, industrial
		16F84A-04	—	1.0	3.0	µA	VDD = 4.0V, WDT disabled, industrial
D021B		16F84A-20	—	1.5	5.5	µA	VDD = 4.5V, WDT disabled, extended
		16F84A-04	—	1.0	5.0	µA	VDD = 4.0V, WDT disabled, extended
		Module Differential Current (Note 5)					
D022	ΔIWDT	Watchdog Timer	—	.20	16	µA	VDD = 2.0V, Industrial, Commercial
			—	3.5	20	µA	VDD = 4.0V, Commercial
			—	3.5	28	µA	VDD = 4.0V, Industrial, Extended
			—	4.8	25	µA	VDD = 4.5V, Commercial
			—	4.8	30	µA	VDD = 4.5V, Industrial, Extended

Legend: Rows with standard voltage device data only are shaded for improved readability.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

NR Not rated for operation.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD,
T0CKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD measurement.

9.2 DC Characteristics: PIC16F84A-04 (Commercial, Industrial) PIC16F84A-20 (Commercial, Industrial) PIC16LF84A-04 (Commercial, Industrial)

DC Characteristics All Pins Except Power Supply Pins			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) Operating voltage VDD range as described in DC specifications (Section 9.1)				
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
	VIL	**Input Low Voltage**					
		I/O ports:					
D030		with TTL buffer	VSS	—	0.8	V	4.5V ≤ VDD ≤ 5.5V **(Note 4)**
D030A			VSS	—	0.16VDD	V	Entire range **(Note 4)**
D031		with Schmitt Trigger buffer	VSS	—	0.2VDD	V	Entire range
D032		$\overline{MCLR}$, RA4/T0CKI	VSS	—	0.2VDD	V	
D033		OSC1 (XT, HS and LP modes)	VSS	—	0.3VDD	V	**(Note 1)**
D034		OSC1 (RC mode)	VSS	—	0.1VDD	V	
	VIH	**Input High Voltage**					
		I/O ports:		—			
D040		with TTL buffer	2.0	—	VDD	V	4.5V ≤ VDD ≤ 5.5V **(Note 4)**
D040A			0.25VDD+0.8	—	VDD	V	Entire range **(Note 4)**
D041		with Schmitt Trigger buffer	0.8 VDD	—	VDD		Entire range
D042		$\overline{MCLR}$,	0.8 VDD	—	VDD	V	
D042A		RA4/T0CKI	0.8 VDD	—	8.5	V	
D043		OSC1 (XT, HS and LP modes)	0.8 VDD	—	VDD	V	**(Note 1)**
D043A		OSC1 (RC mode)	0.9 VDD		VDD	V	
D050	VHYS	**Hysteresis of Schmitt Trigger Inputs**	—	0.1	—	V	
D070	IPURB	**PORTB Weak Pull-up Current**	50	250	400	µA	VDD = 5.0V, VPIN = VSS
	IIL	**Input Leakage Current (Notes 2, 3)**					
D060		I/O ports	—	—	±1	µA	Vss ≤ VPIN ≤ VDD, Pin at hi-impedance
D061		$\overline{MCLR}$, RA4/T0CKI	—	—	±5	µA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	—	±5	µA	Vss ≤ VPIN ≤ VDD, XT, HS and LP osc configuration

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. Do not drive the PIC16F84A with an external clock while the device is in RC mode, or chip damage may result.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: The user may choose the better of the two specs.

PIC16F84A

9.2 DC Characteristics: PIC16F84A-04 (Commercial, Industrial) PIC16F84A-20 (Commercial, Industrial) PIC16LF84A-04 (Commercial, Industrial) (Continued)

DC Characteristics All Pins Except Power Supply Pins			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) Operating voltage VDD range as described in DC specifications (Section 9.1)				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VOL	**Output Low Voltage**					
D080		I/O ports	—	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V
D083		OSC2/CLKOUT	—	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, (RC mode only)
	VOH	**Output High Voltage**					
D090		I/O ports (Note 3)	VDD-0.7	—	—	V	IOH = -3.0 mA, VDD = 4.5V
D092		OSC2/CLKOUT (Note 3)	VDD-0.7	—	—	V	IOH = -1.3 mA, VDD = 4.5V (RC mode only)
	VOD	**Open Drain High Voltage**					
D150		RA4 pin	—	—	8.5	V	
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2 pin	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	CIO	All I/O pins and OSC2 (RC mode)	—	—	50	pF	
		Data EEPROM Memory					
D120	ED	Endurance	1M	10M	—	E/W	25°C at 5V
D121	VDRW	VDD for read/write	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D122	TDEW	Erase/Write cycle time	—	4	8	ms	
		Program FLASH Memory					
D130	EP	Endurance	1000	10K	—	E/W	
D131	VPR	VDD for read	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D132	VPEW	VDD for erase/write	4.5	—	5.5	V	
D133	TPEW	Erase/Write cycle time	—	4	8	ms	

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. Do not drive the PIC16F84A with an external clock while the device is in RC mode, or chip damage may result.

2: The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: The user may choose the better of the two specs.

9.3 AC (Timing) Characteristics

9.3.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
2	to	os, osc	OSC1
ck	CLKOUT	ost	oscillator start-up timer
cy	cycle time	pwrt	power-up timer
io	I/O port	rbt	RBx pins
inp	INT pin	t0	T0CKI
mp	$\overline{\text{MCLR}}$	wdt	watchdog timer

Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (high impedance)	V	Valid
L	Low	Z	High Impedance

PIC16F84A

9.3.2 TIMING CONDITIONS

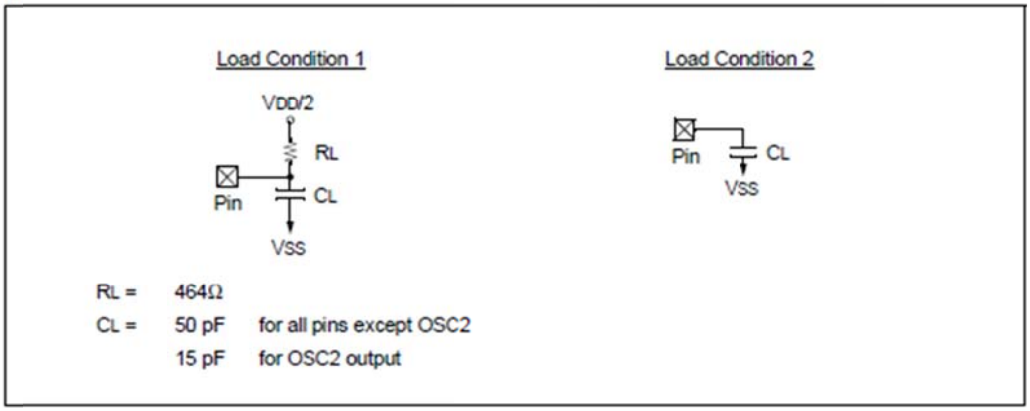
The temperature and voltages specified in Table 9-1 apply to all timing specifications unless otherwise noted. All timings are measured between high and low measurement points as indicated in Figure 9-4. Figure 9-5 specifies the load conditions for the timing specifications.

TABLE 9-1: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

AC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial -40°C ≤ TA ≤ +85°C for industrial Operating voltage VDD range as described in DC specifications (Section 9.1)

FIGURE 9-4: PARAMETER MEASUREMENT INFORMATION

FIGURE 9-5: LOAD CONDITIONS



9.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 9-6: EXTERNAL CLOCK TIMING

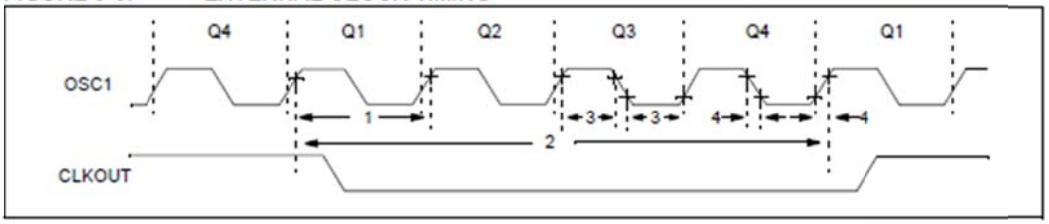


TABLE 9-2: EXTERNAL CLOCK TIMING REQUIREMENTS

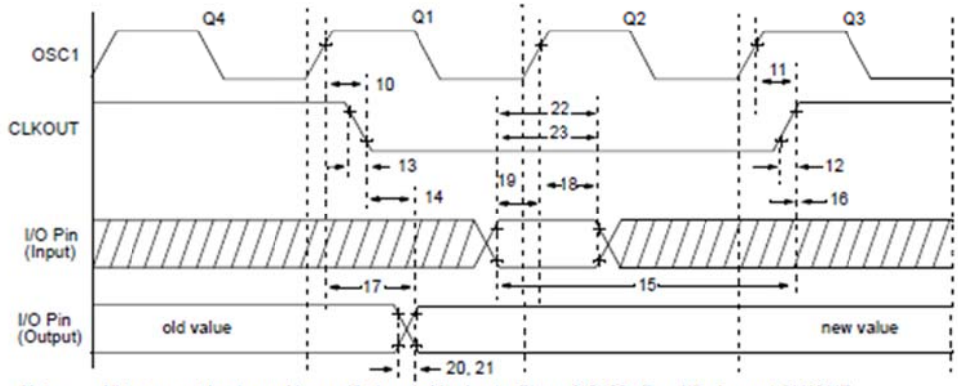
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions	
	FOSC	External CLKIN Frequency(1)	DC	—	2	MHz	XT, RC osc	(-04, LF)
			DC	—	4	MHz	XT, RC osc	(-04)
			DC	—	20	MHz	HS osc	(-20)
			DC	—	200	kHz	LP osc	(-04, LF)
		Oscillator Frequency(1)	DC	—	2	MHz	RC osc	(-04, LF)
			DC	—	4	MHz	RC osc	(-04)
			0.1	—	2	MHz	XT osc	(-04, LF)
			0.1	—	4	MHz	XT osc	(-04)
			1.0	—	20	MHz	HS osc	(-20)
			DC	—	200	kHz	LP osc	(-04, LF)
1	TOSC	External CLKIN Period(1)	500	—	—	ns	XT, RC osc	(-04, LF)
			250	—	—	ns	XT, RC osc	(-04)
			50	—	—	ns	HS osc	(-20)
			5.0	—	—	µs	LP osc	(-04, LF)
		Oscillator Period(1)	500	—	—	ns	RC osc	(-04, LF)
			250	—	—	ns	RC osc	(-04)
			500	—	10,000	ns	XT osc	(-04, LF)
			250	—	10,000	ns	XT osc	(-04)
			50	—	1,000	ns	HS osc	(-20)
			5.0	—	—	µs	LP osc	(-04, LF)
2	TCY	Instruction Cycle Time(1)	0.2	4/FOSC	DC	µs		
3	TosL, TosH	Clock in (OSC1) High or Low Time	60	—	—	ns	XT osc	(-04, LF)
			50	—	—	ns	XT osc	(-04)
			2.0	—	—	µs	LP osc	(-04, LF)
			17.5	—	—	ns	HS osc	(-20)
4	TosR, TosF	Clock in (OSC1) Rise or Fall Time	25	—	—	ns	XT osc	(-04)
			50	—	—	ns	LP osc	(-04, LF)
			7.5	—	—	ns	HS osc	(-20)

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "Min." values with an external clock applied to the OSC1 pin.
When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

PIC16F84A

FIGURE 9-7: CLKOUT AND I/O TIMING



Note: All tests must be done with specified capacitive loads (Figure 9-5) 50 pF on I/O pins and CLKOUT.

TABLE 9-3: CLKOUT AND I/O TIMING REQUIREMENTS

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKOUT↓	Standard	—	15	30	ns	(Note 1)
10A			Extended (LF)	—	15	120	ns	(Note 1)
11	TosH2ckH	OSC1↑ to CLKOUT↑	Standard	—	15	30	ns	(Note 1)
11A			Extended (LF)	—	15	120	ns	(Note 1)
12	TckR	CLKOUT rise time	Standard	—	15	30	ns	(Note 1)
12A			Extended (LF)	—	15	100	ns	(Note 1)
13	TckF	CLKOUT fall time	Standard	—	15	30	ns	(Note 1)
13A			Extended (LF)	—	15	100	ns	(Note 1)
14	TckL2ioV	CLKOUT ↓ to Port out valid		—	—	0.5TCY +20	ns	(Note 1)
15	TioV2ckH	Port in valid before CLKOUT ↑	Standard	0.30TCY + 30	—	—	ns	(Note 1)
			Extended (LF)	0.30TCY + 80	—	—	ns	(Note 1)
16	TckH2ioI	Port in hold after CLKOUT ↑		0	—	—	ns	(Note 1)
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	Standard	—	—	125	ns	
			Extended (LF)	—	—	250	ns	
18	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	Standard	10	—	—	ns	
			Extended (LF)	10	—	—	ns	
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	Standard	-75	—	—	ns	
			Extended (LF)	-175	—	—	ns	
20	TioR	Port output rise time	Standard	—	10	35	ns	
20A			Extended (LF)	—	10	70	ns	
21	TioF	Port output fall time	Standard	—	10	35	ns	
21A			Extended (LF)	—	10	70	ns	
22	TINP	INT pin high or low time	Standard	20	—	—	ns	
22A			Extended (LF)	55	—	—	ns	
23	TRBP	RB7:RB4 change INT high or low time	Standard	TOSC§	—	—	ns	
23A			Extended (LF)	TOSC§	—	—	ns	

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ By design.

Note 1: Measurements are taken in RC mode where CLKOUT output is 4 x TOSC.

FIGURE 9-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

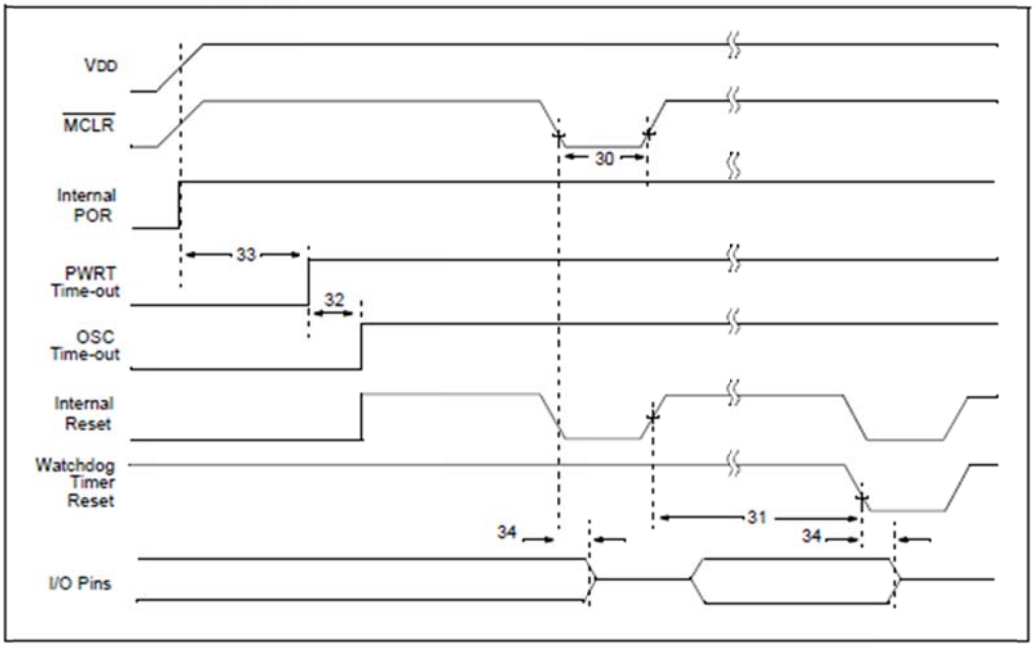


TABLE 9-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{MCLR}$ Pulse Width (low)	2	—	—	µs	VDD = 5.0V
31	TWDT	Watchdog Timer Time-out Period (No Prescaler)	7	18	33	ms	VDD = 5.0V
32	TOST	Oscillation Start-up Timer Period		1024TOSC		ms	TOSC = OSC1 period
33	TPWRT	Power-up Timer Period	28	72	132	ms	VDD = 5.0V
34	TIOZ	I/O hi-impedance from $\overline{MCLR}$ Low or RESET	—	—	100	ns	

† Data in "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

PIC16F84A

FIGURE 9-9: TIMER0 CLOCK TIMINGS

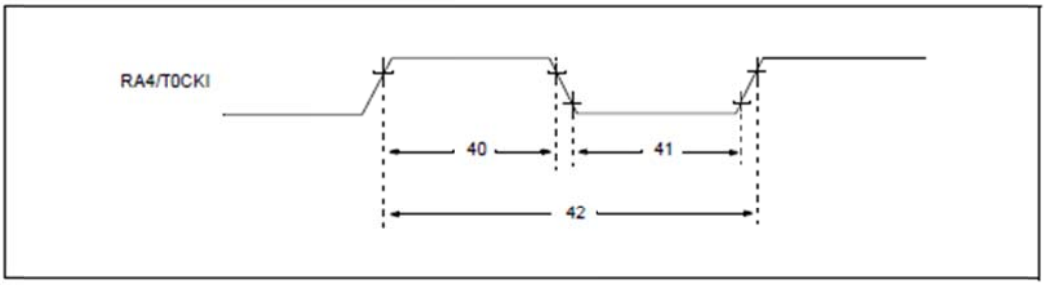


TABLE 9-5: TIMER0 CLOCK REQUIREMENTS

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	50	—	—	ns	2.0V ≤ VDD ≤ 3.0V
				30	—	—	ns	3.0V ≤ VDD ≤ 6.0V
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	50	—	—	ns	2.0V ≤ VDD ≤ 3.0V
				20	—	—	ns	3.0V ≤ VDD ≤ 6.0V
42	Tt0P	T0CKI Period		$\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value (2, 4, ..., 256)

† Data in "Typ" column is at 5.0V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

10.0 DC/AC CHARACTERISTIC GRAPHS

The graphs provided in this section are for **design guidance** and are **not tested**.

In some graphs, the data presented are **outside specified operating range** (i.e., outside specified VDD range). This is for **information only** and devices are ensured to operate properly only within the specified range.

The data presented in this section is a **statistical summary** of data collected on units from different lots over a period of time and matrix samples. 'Typical' represents the mean of the distribution at 25°C. 'Max' or 'Min' represents (mean + 3σ) or (mean - 3σ), respectively, where σ is a standard deviation over the whole temperature range.

PIC16F84A

FIGURE 10-1: TYPICAL IDD vs. FOSC OVER VDD (HS MODE, 25°C)

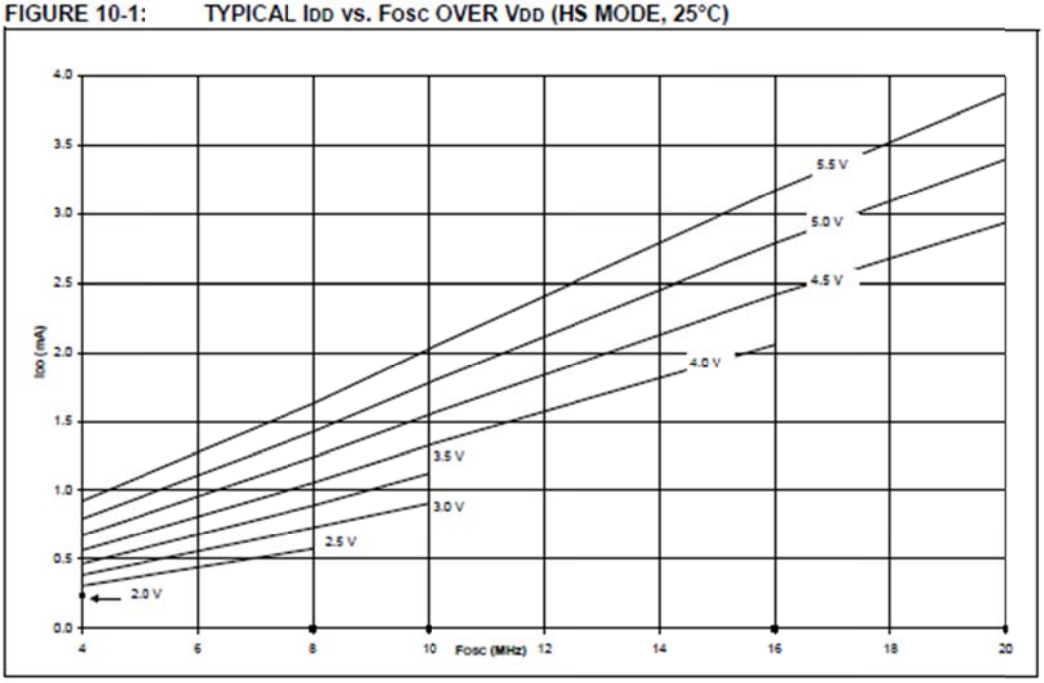


FIGURE 10-2: MAXIMUM IDD vs. FOSC OVER VDD (HS MODE, -40° TO +125°C)

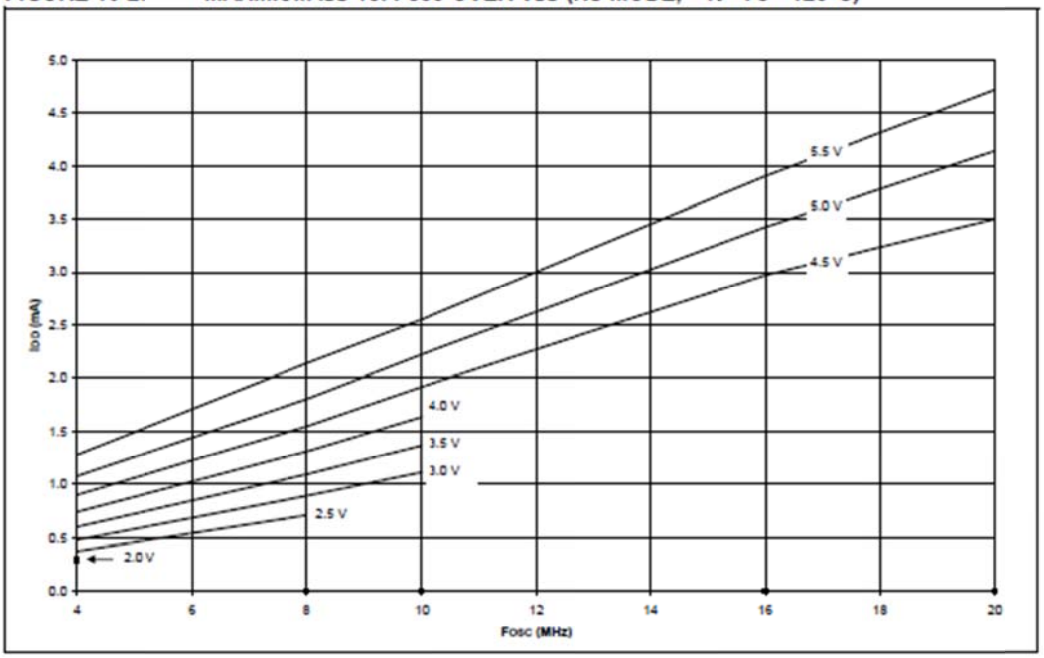


FIGURE 10-3: TYPICAL IDD vs. FOSC OVER VDD (XT MODE, 25°C)

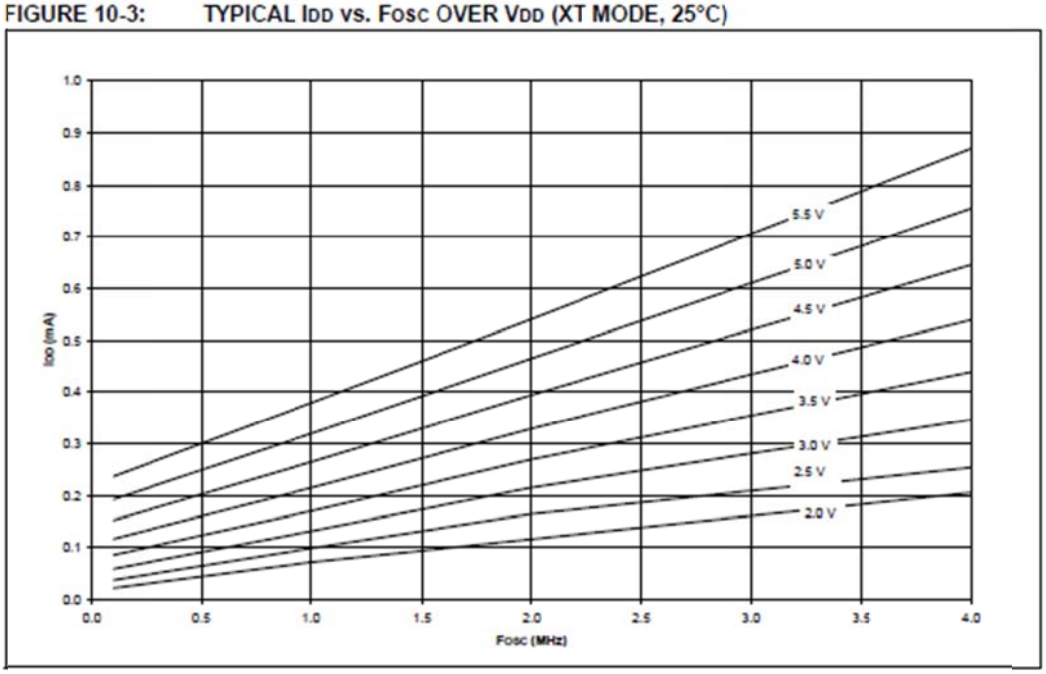
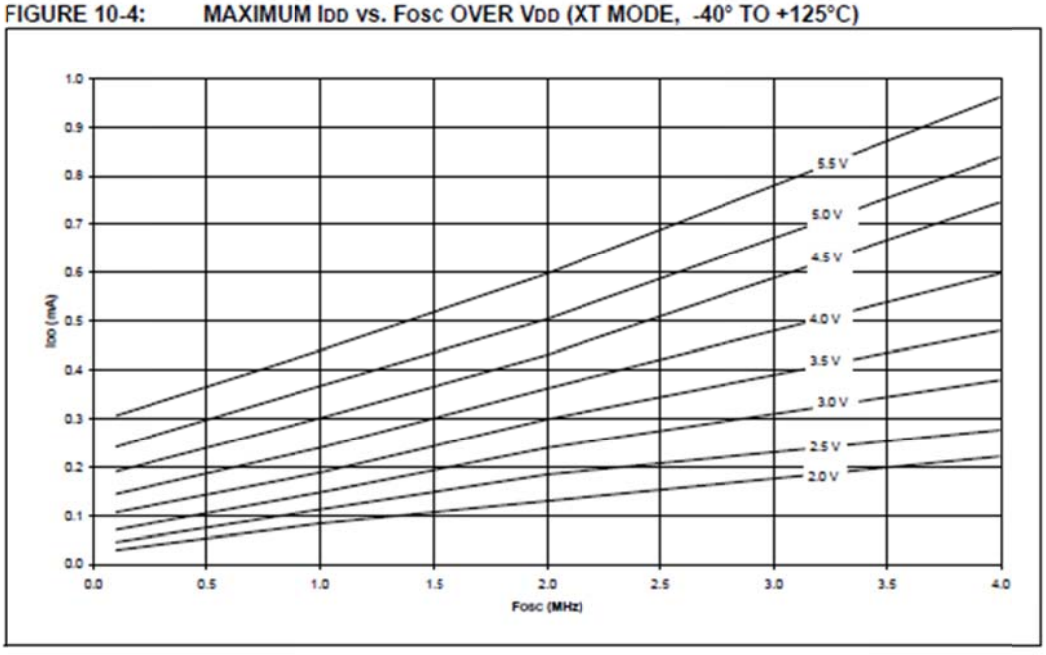


FIGURE 10-4: MAXIMUM IDD vs. FOSC OVER VDD (XT MODE, -40° TO +125°C)



PIC16F84A

FIGURE 10-5: TYPICAL IDD vs. FOSC OVER VDD (LP MODE, 25°C)

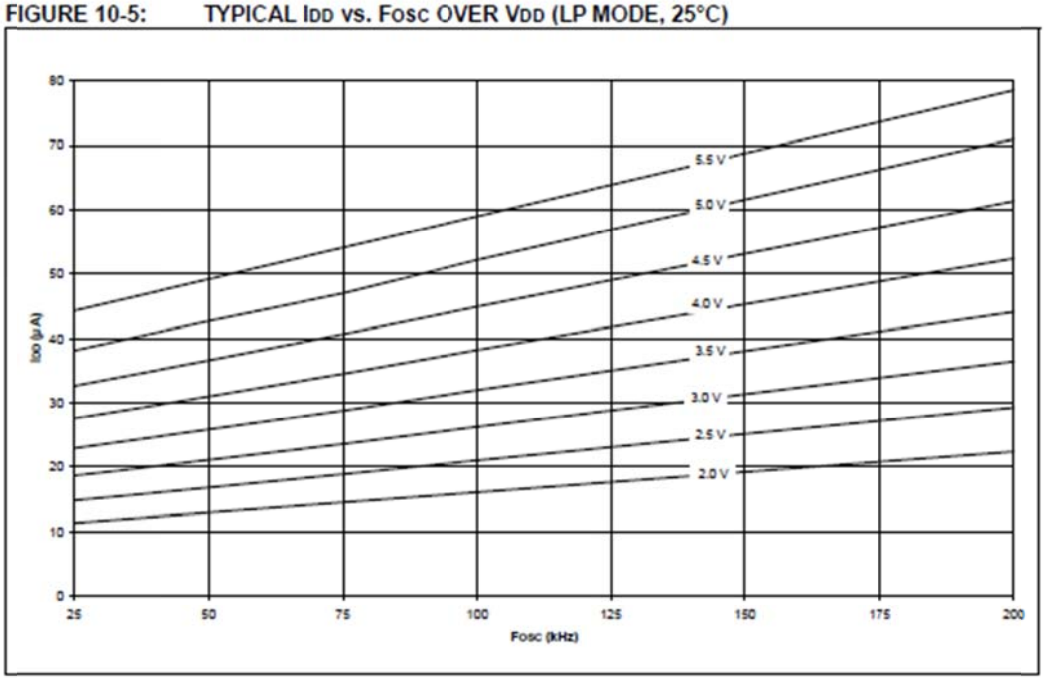


FIGURE 10-6: MAXIMUM IDD vs. FOSC OVER VDD (LP MODE, -40° TO +125°C)

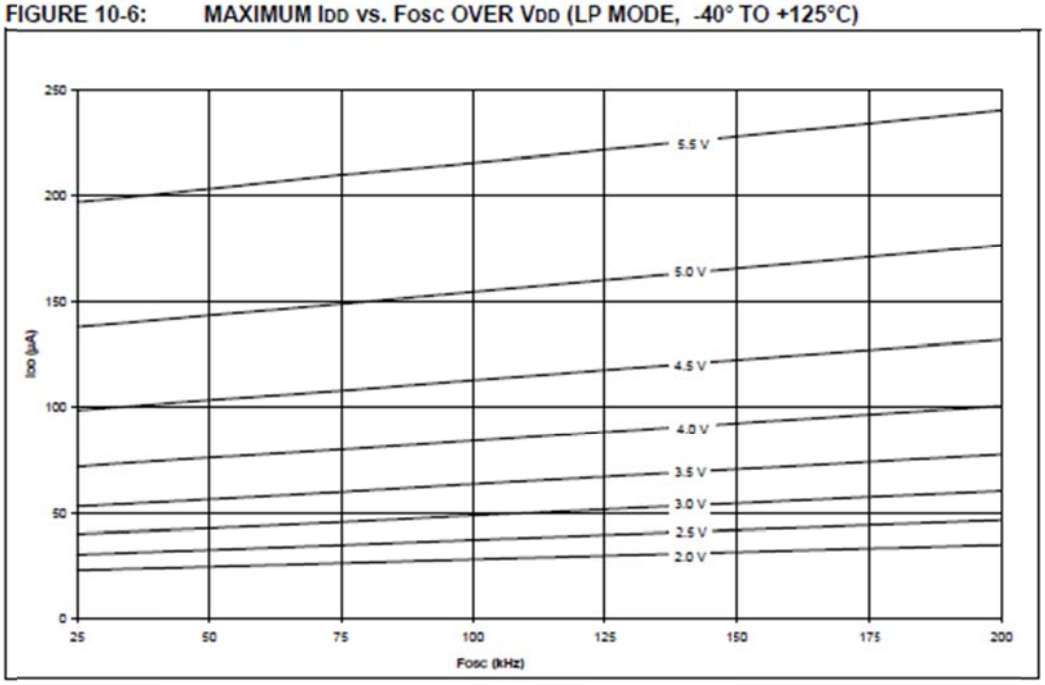


FIGURE 10-7: AVERAGE Fosc vs. VDD FOR R (RC MODE, C = 22 pF, 25°C)

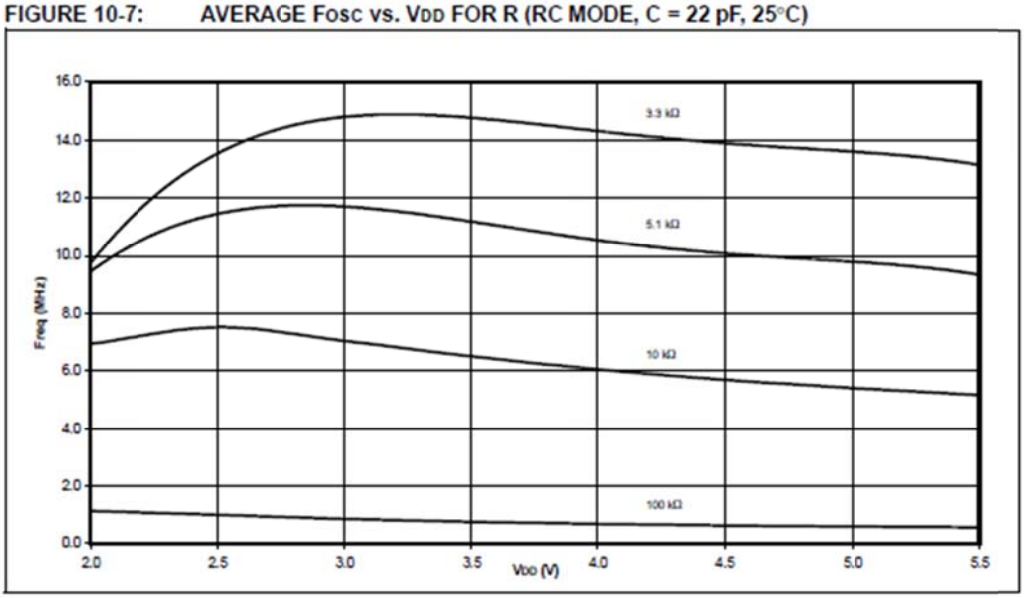
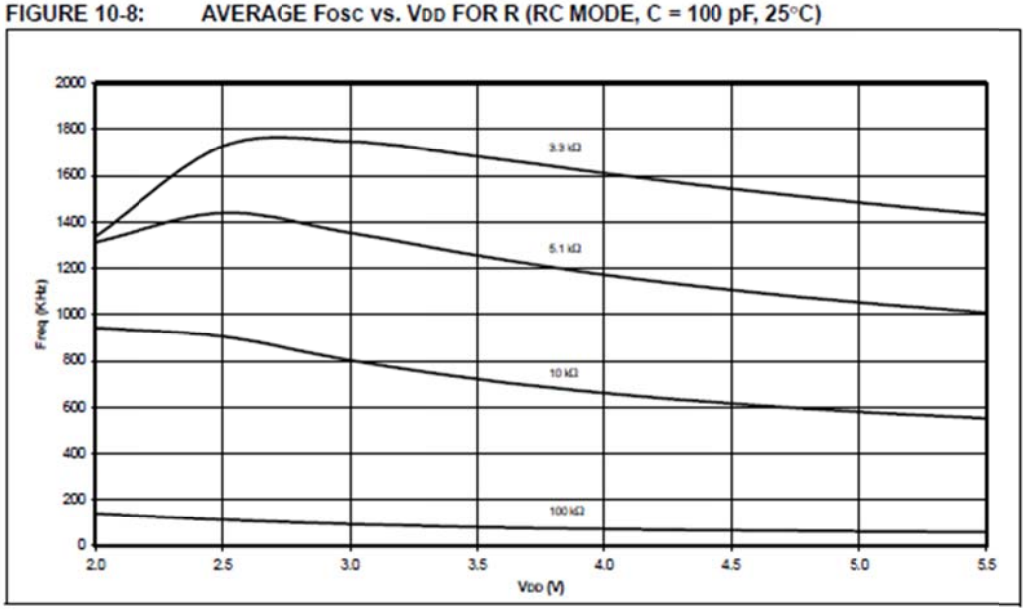


FIGURE 10-8: AVERAGE Fosc vs. VDD FOR R (RC MODE, C = 100 pF, 25°C)



PIC16F84A

FIGURE 10-9: AVERAGE Fosc vs. VDD FOR R (RC MODE, C = 300 pF, 25°C)

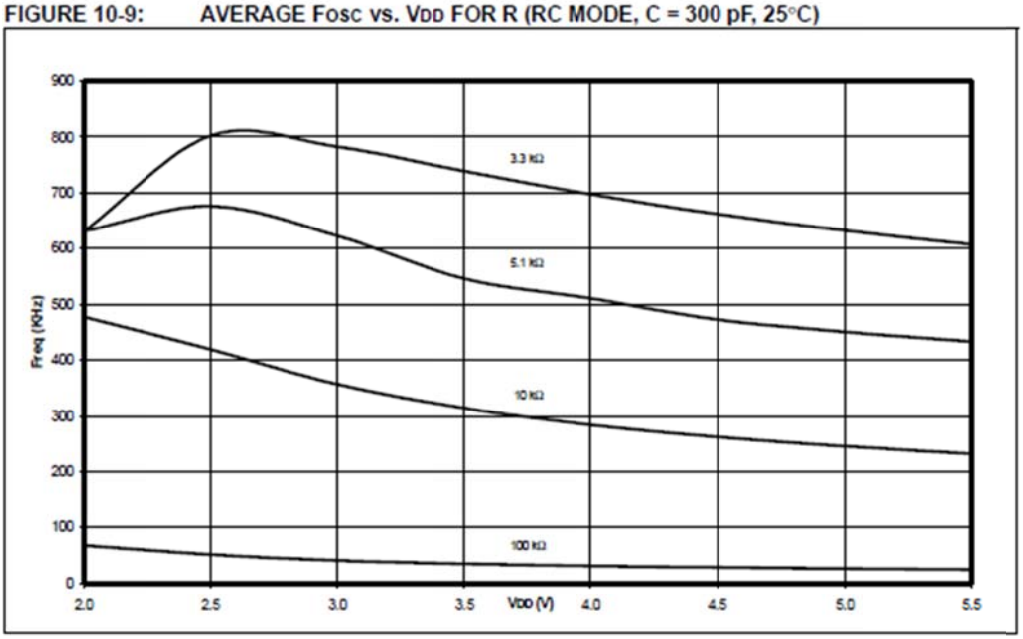


FIGURE 10-10: IPD vs. VDD (SLEEP MODE, ALL PERIPHERALS DISABLED)

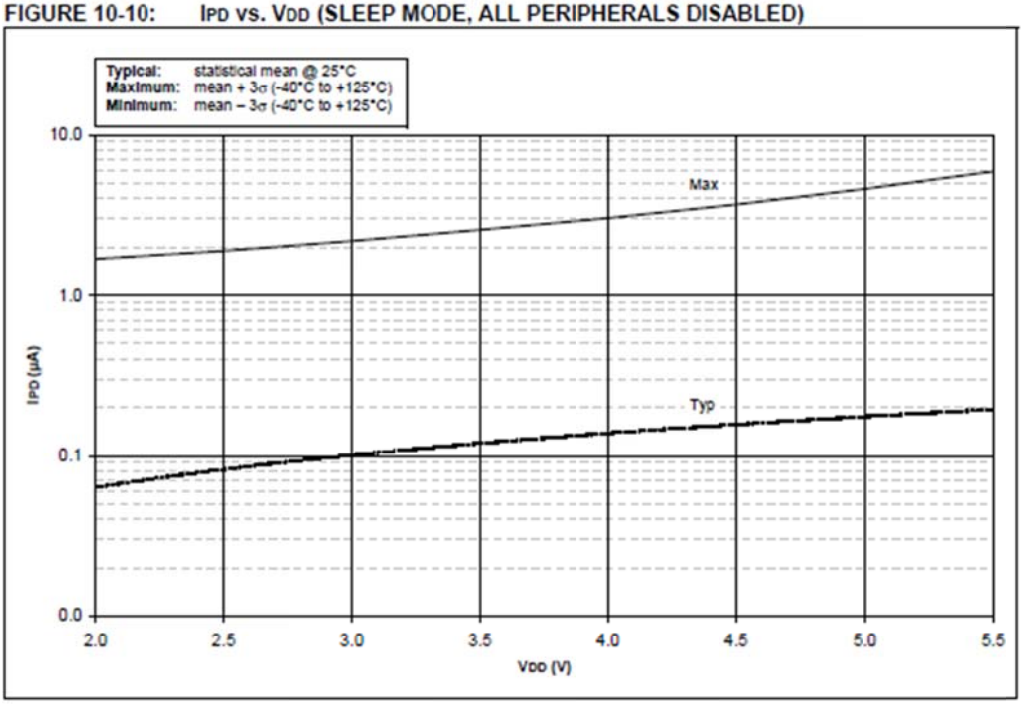


FIGURE 10-11: IPD vs. VDD (WDT MODE)

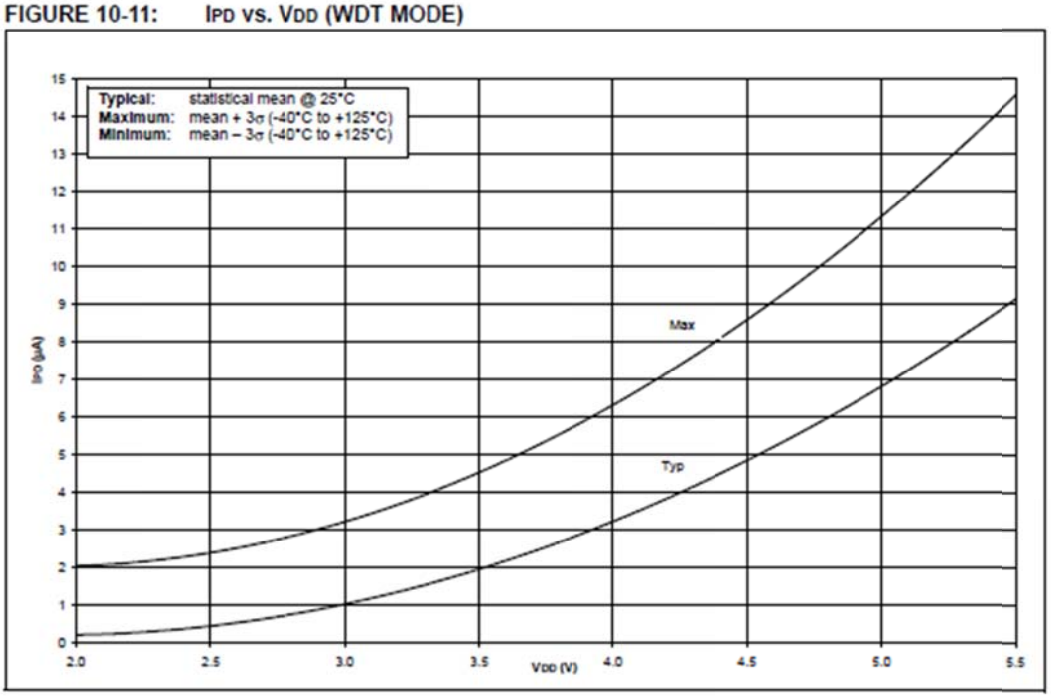
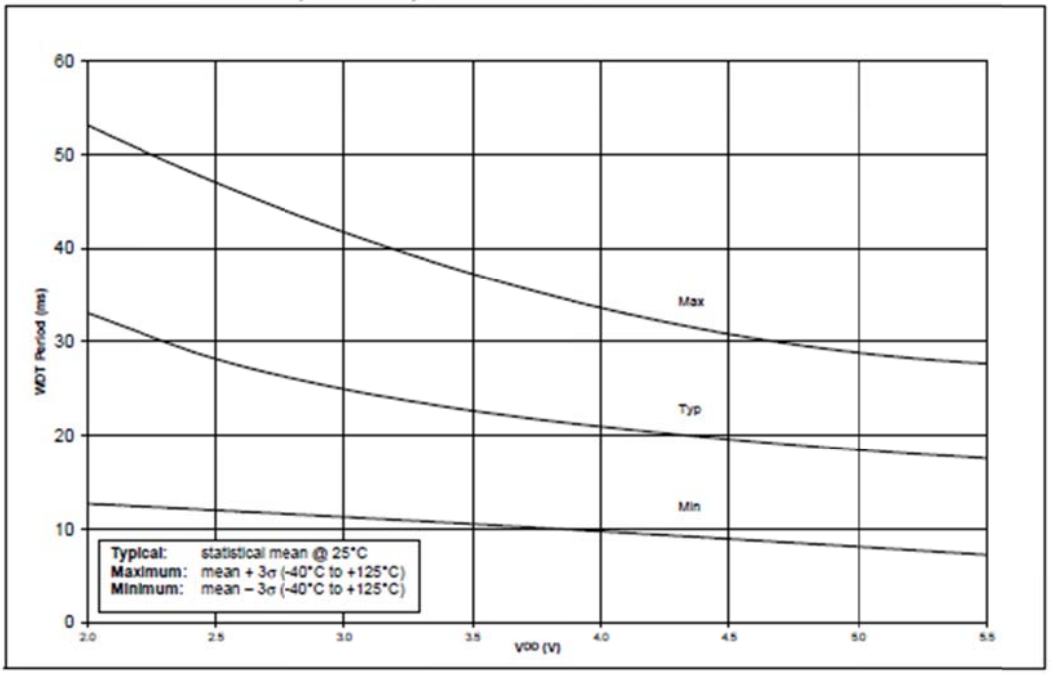


FIGURE 10-12: TYPICAL, MINIMUM, AND MAXIMUM WDT PERIOD vs. VDD OVER TEMP



PIC16F84A

FIGURE 10-13: TYPICAL, MINIMUM AND MAXIMUM V_{OH} vs. I_{OH} (V_{DD} = 5V, -40°C TO +125°C)

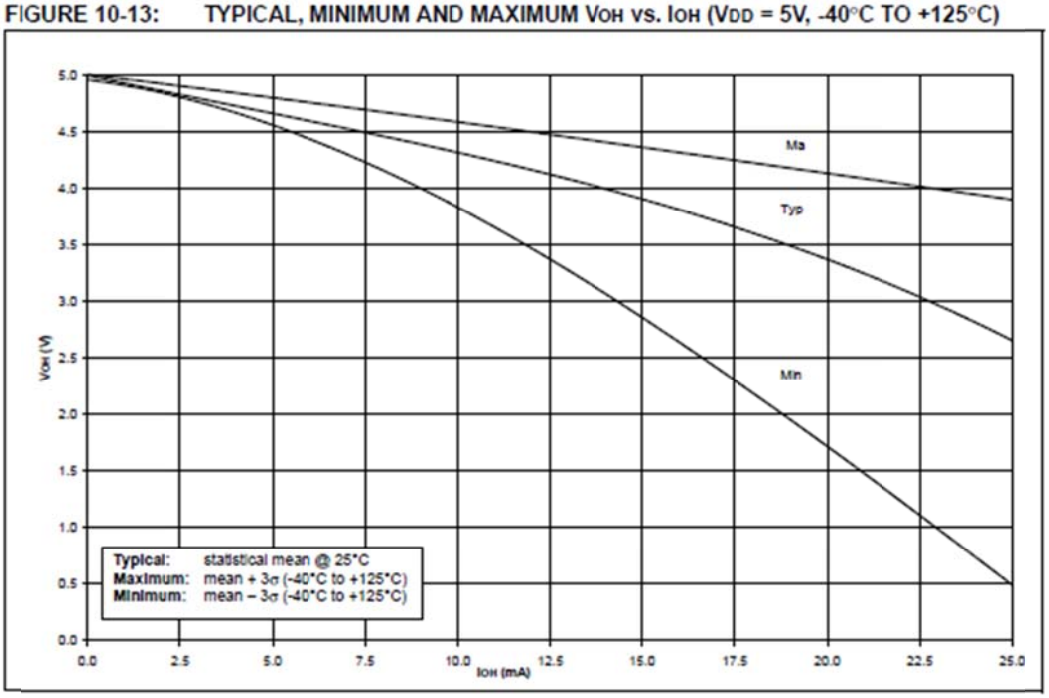


FIGURE 10-14: TYPICAL, MINIMUM AND MAXIMUM V_{OH} vs. I_{OH} (V_{DD} = 3V, -40°C TO +125°C)

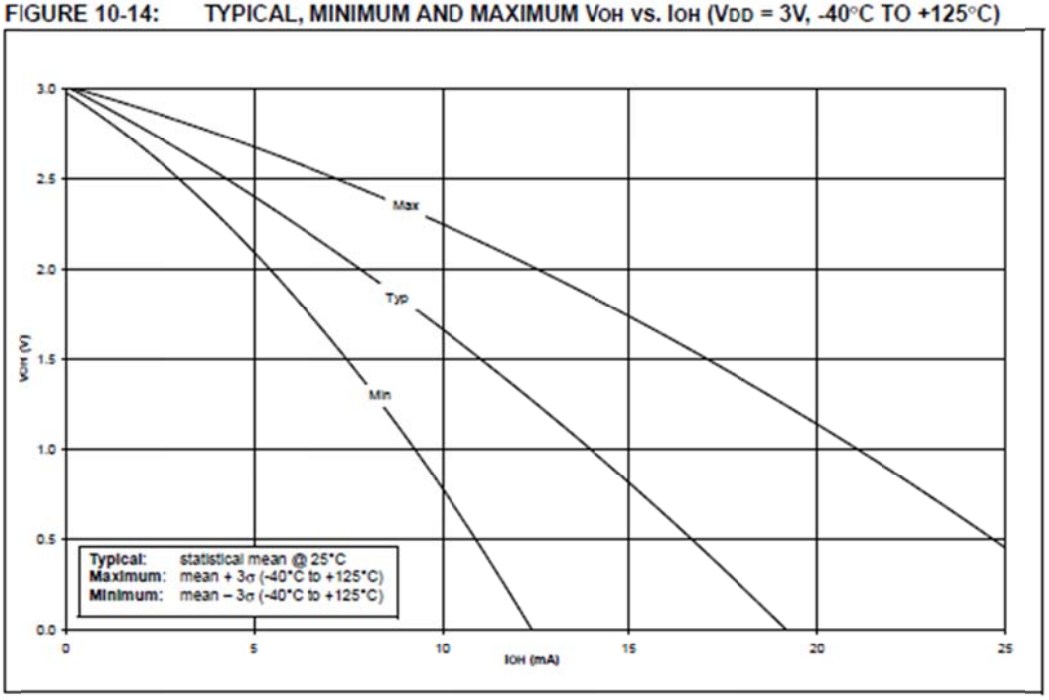


FIGURE 10-15: TYPICAL, MINIMUM AND MAXIMUM VOL vs. IOL (VDD = 5V, -40°C TO +125°C)

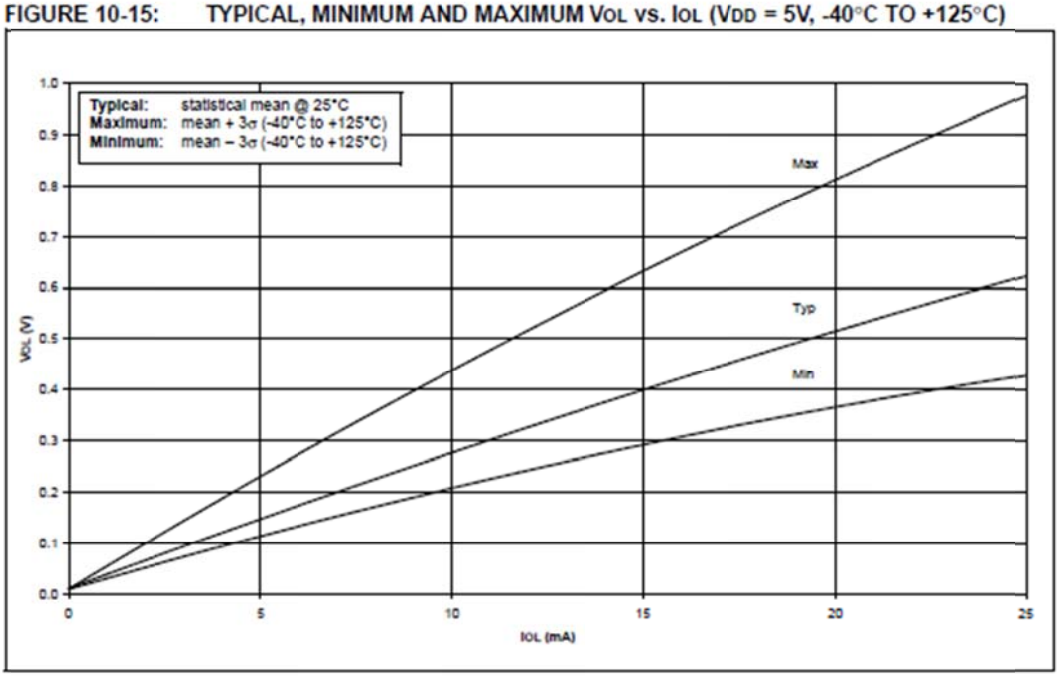
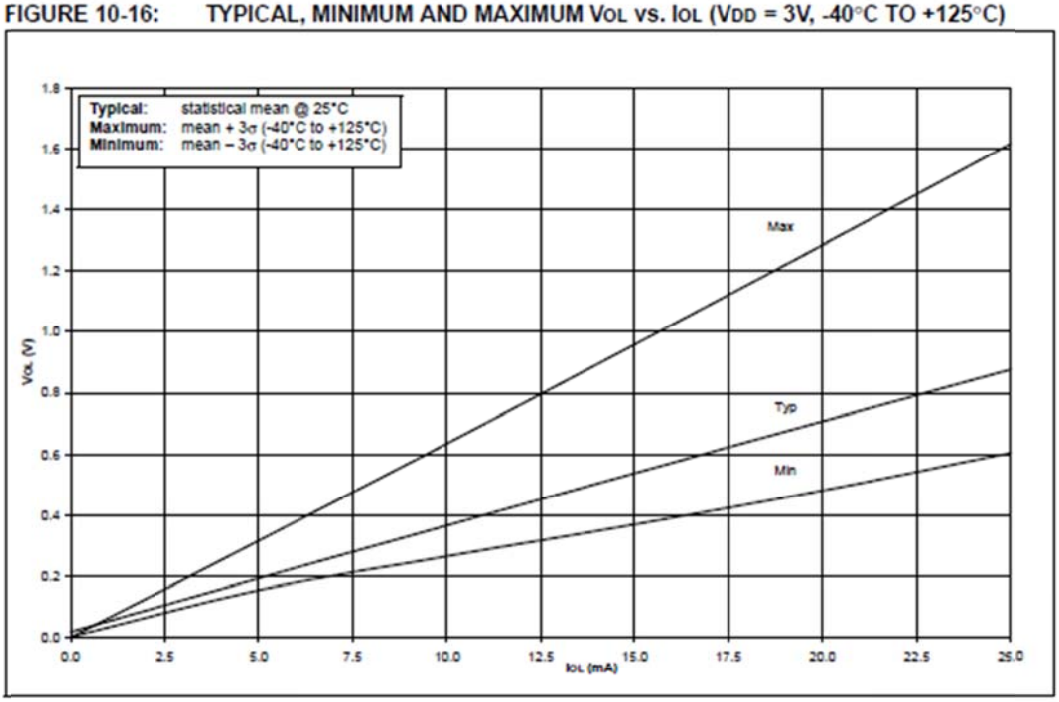


FIGURE 10-16: TYPICAL, MINIMUM AND MAXIMUM VOL vs. IOL (VDD = 3V, -40°C TO +125°C)



PIC16F84A

FIGURE 10-17: MINIMUM AND MAXIMUM VIN vs. VDD, (TTL INPUT, -40°C TO +125°C)

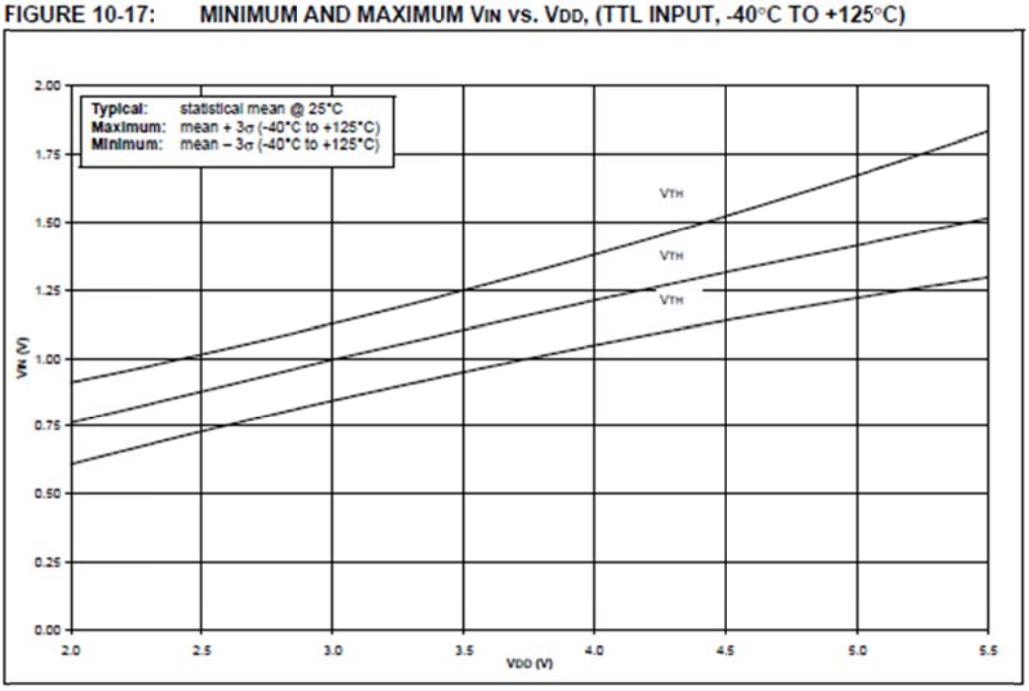
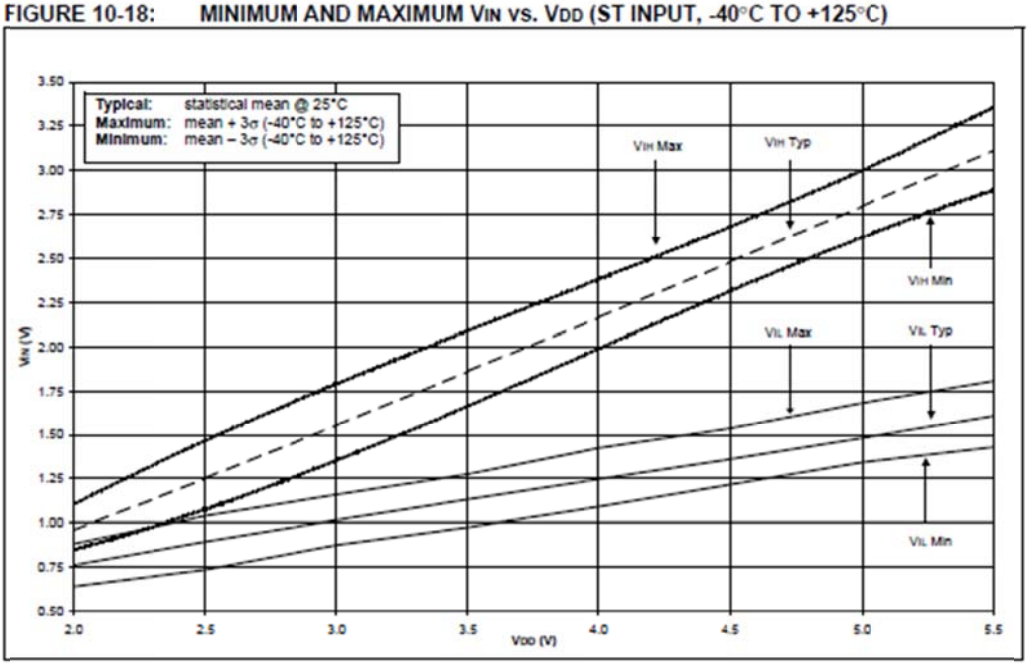


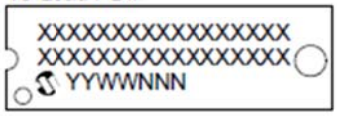
FIGURE 10-18: MINIMUM AND MAXIMUM VIN vs. VDD (ST INPUT, -40°C TO +125°C)



11.0 PACKAGING INFORMATION

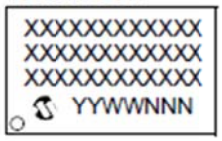
11.1 Package Marking Information

18-Lead PDIP

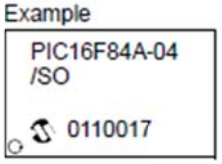


Example

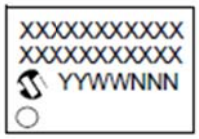
18-Lead SOIC



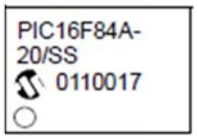
Example



20-Lead SSOP



Example



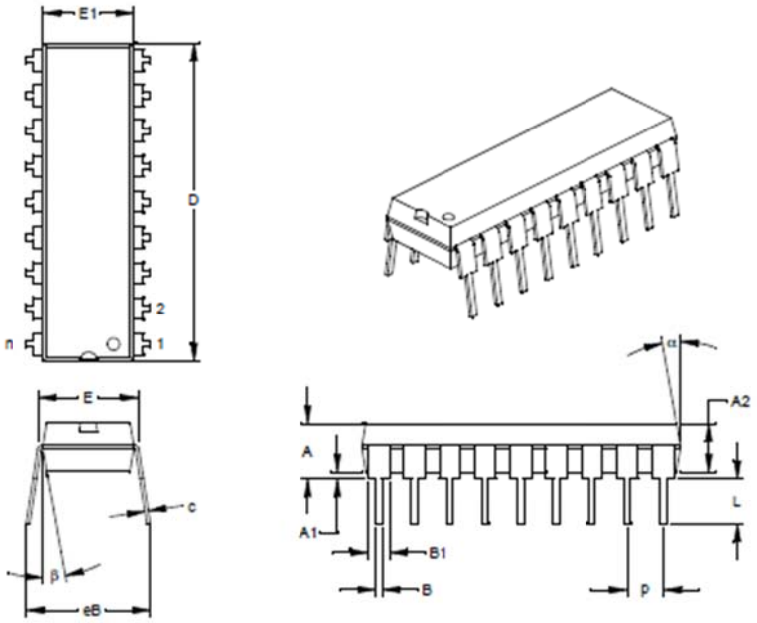
Legend:	XX...X	Customer specific information*
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard PICmicro device marking consists of Microchip part number, year code, week code, and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip sales office. For QTP devices, any special marking adders are included in QTP price.

PIC16F84A

18-Lead Plastic Dual In-line (P) – 300 mil (PDIP)

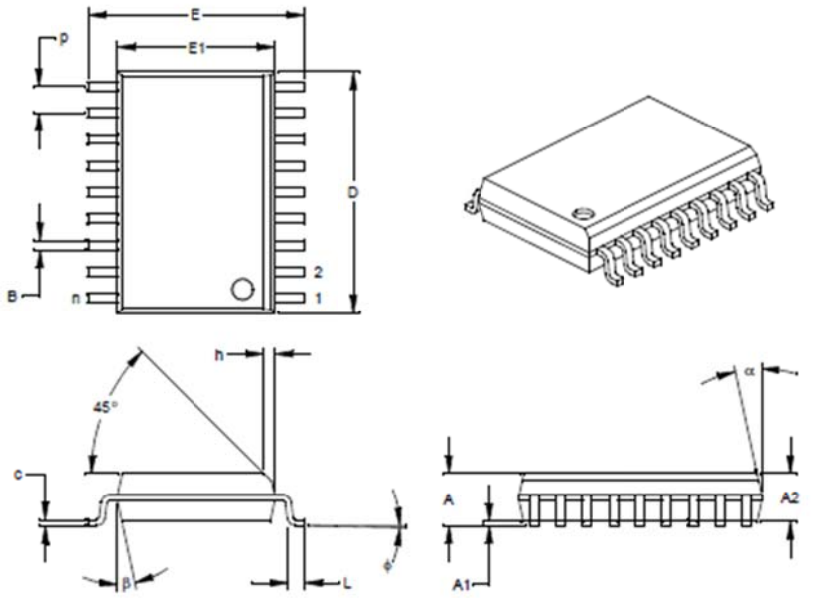


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.155	.170	3.56	3.94	4.32
Molded Package Thickness	A2	.115	.130	.145	2.92	3.30	3.68
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.313	.325	7.62	7.94	8.26
Molded Package Width	E1	.240	.250	.260	6.10	6.35	6.60
Overall Length	D	.890	.898	.905	22.61	22.80	22.99
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.045	.058	.070	1.14	1.46	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.310	.370	.430	7.87	9.40	10.92
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-001
Drawing No. C04-007

18-Lead Plastic Small Outline (SO) – Wide, 300 mil (SOIC)



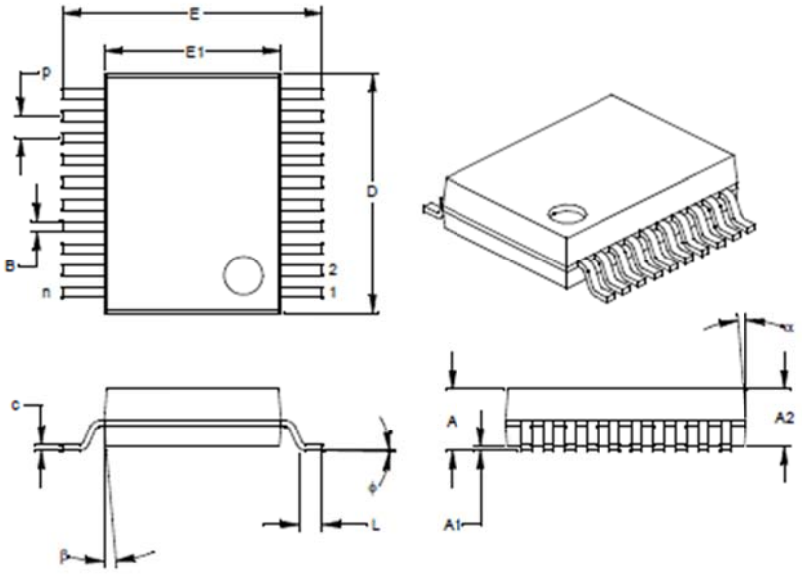
Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.291	.295	.299	7.39	7.49	7.59
Overall Length	D	.446	.454	.462	11.33	11.53	11.73
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle	φ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.012	0.23	0.27	0.30
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-013
Drawing No. C04-051

PIC16F84A

20-Lead Plastic Shrink Small Outline (SS) – 209 mil, 5.30 mm (SSOP)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		20			20	
Pitch	p		.026			0.65	
Overall Height	A	.068	.073	.078	1.73	1.85	1.98
Molded Package Thickness	A2	.064	.068	.072	1.63	1.73	1.83
Standoff §	A1	.002	.006	.010	0.05	0.15	0.25
Overall Width	E	.299	.309	.322	7.59	7.85	8.18
Molded Package Width	E1	.201	.207	.212	5.11	5.25	5.38
Overall Length	D	.278	.284	.289	7.06	7.20	7.34
Foot Length	L	.022	.030	.037	0.56	0.75	0.94
Lead Thickness	c	.004	.007	.010	0.10	0.18	0.25
Foot Angle	φ	0	4	8	0.00	101.60	203.20
Lead Width	B	.010	.013	.015	0.25	0.32	0.38
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-150
Drawing No. C04-072

APPENDIX A: REVISION HISTORY

Version	Date	Revision Description
A	9/98	This is a new data sheet. However, the devices described in this data sheet are the upgrades to the devices found in the *PIC16F8X Data Sheet*, DS30430.
B	8/01	Added DC and AC Characteristics Graphs and Tables to Section 10.

PIC16F84A

APPENDIX B: CONVERSION CONSIDERATIONS

Considerations for converting from one PIC16X8X device to another are listed in Table 1.

TABLE 1: CONVERSION CONSIDERATIONS - PIC16C84, PIC16F83/F84, PIC16CR83/CR84, PIC16F84A

Difference	PIC16C84	PIC16F83/F84	PIC16CR83/ CR84	PIC16F84A
Program Memory Size	1K x 14	512 x 14 / 1K x 14	512 x 14 / 1K x 14	1K x 14
Data Memory Size	36 x 8	36 x 8 / 68 x 8	36 x 8 / 68 x 8	68 x 8
Voltage Range	2.0V - 6.0V (-40°C to +85°C)	2.0V - 6.0V (-40°C to +85°C)	2.0V - 6.0V (-40°C to +85°C)	2.0V - 5.5V (-40°C to +125°C)
Maximum Operating Frequency	10 MHz	10 MHz	10 MHz	20 MHz
Supply Current (I_{DD}). See parameter # D014 in the electrical specs for more detail.	I_{DD} (typ) = 60 µA I_{DD} (max) = 400 µA (LP osc, F_{OSC} = 32 kHz, V_{DD} = 2.0V, WDT disabled)	I_{DD} (typ) = 15 µA I_{DD} (max) = 45 µA (LP osc, F_{OSC} = 32 kHz, V_{DD} = 2.0V, WDT disabled)	I_{DD} (typ) = 15 µA I_{DD} (max) = 45 µA (LP osc, F_{OSC} = 32 kHz, V_{DD} = 2.0V, WDT disabled)	I_{DD} (typ) = 15 µA I_{DD} (max) = 45 µA (LP osc, F_{OSC} = 32 kHz, V_{DD} = 2.0V, WDT disabled)
Power-down Current (I_{PD}). See parameters # D020, D021, and D021A in the electrical specs for more detail.	I_{PD} (typ) = 26 µA I_{PD} (max) = 100 µA (V_{DD} = 2.0V, WDT disabled, industrial)	I_{PD} (typ) = 0.4 µA I_{PD} (max) = 9 µA (V_{DD} = 2.0V, WDT disabled, industrial)	I_{PD} (typ) = 0.4 µA I_{PD} (max) = 6 µA (V_{DD} = 2.0V, WDT disabled, industrial)	I_{PD} (typ) = 0.4 µA I_{PD} (max) = 1 µA (V_{DD} = 2.0V, WDT disabled, industrial)
Input Low Voltage (V_{IL}). See parameters # D032 and D034 in the electrical specs for more detail.	V_{IL} (max) = 0.2V_{DD} (OSC1, RC mode)	V_{IL} (max) = 0.1V_{DD} (OSC1, RC mode)	V_{IL} (max) = 0.1V_{DD} (OSC1, RC mode)	V_{IL} (max) = 0.1V_{DD} (OSC1, RC mode)
Input High Voltage (V_{IH}). See parameter # D040 in the electrical specs for more detail.	V_{IH} (min) = 0.36V_{DD} (I/O Ports with TTL, 4.5V ≤ V_{DD} ≤ 5.5V)	V_{IH} (min) = 2.4V (I/O Ports with TTL, 4.5V ≤ V_{DD} ≤ 5.5V)	V_{IH} (min) = 2.4V (I/O Ports with TTL, 4.5V ≤ V_{DD} ≤ 5.5V)	V_{IH} (min) = 2.4V (I/O Ports with TTL, 4.5V ≤ V_{DD} ≤ 5.5V)
Data EEPROM Memory Erase/Write cycle time (T_{DEW}). See parameter # D122 in the electrical specs for more detail.	T_{DEW} (typ) = 10 ms T_{DEW} (max) = 20 ms	T_{DEW} (typ) = 10 ms T_{DEW} (max) = 20 ms	T_{DEW} (typ) = 10 ms T_{DEW} (max) = 20 ms	T_{DEW} (typ) = 4 ms T_{DEW} (max) = 8 ms
Port Output Rise/Fall time (T_{ioR}, T_{ioF}). See parameters #20, 20A, 21, and 21A in the electrical specs for more detail.	T_{ioR}, T_{ioF} (max) = 25 ns (C84) T_{ioR}, T_{ioF} (max) = 60 ns (LC84)	T_{ioR}, T_{ioF} (max) = 35 ns (C84) T_{ioR}, T_{ioF} (max) = 70 ns (LC84)	T_{ioR}, T_{ioF} (max) = 35 ns (C84) T_{ioR}, T_{ioF} (max) = 70 ns (LC84)	T_{ioR}, T_{ioF} (max) = 35 ns (C84) T_{ioR}, T_{ioF} (max) = 70 ns (LC84)
$\overline{MCLR}$ on-chip filter. See parameter #30 in the electrical specs for more detail.	No	Yes	Yes	Yes
PORTA and crystal oscillator values less than 500 kHz	For crystal oscillator configurations operating below 500 kHz, the device may generate a spurious internal Q-clock when PORTA<0> switches state.	N/A	N/A	N/A
RB0/INT pin	TTL	TTL/ST* (*Schmitt Trigger)	TTL/ST* (*Schmitt Trigger)	TTL/ST* (*Schmitt Trigger)

TABLE 1: CONVERSION CONSIDERATIONS - PIC16C84, PIC16F83/F84, PIC16CR83/CR84, PIC16F84A (CONTINUED)

Difference	PIC16C84	PIC16F83/F84	PIC16CR83/ CR84	PIC16F84A
EEADR<7:6> and IDD	It is recommended that the EEADR<7:6> bits be cleared. When either of these bits is set, the maximum IDD for the device is higher than when both are cleared.	N/A	N/A	N/A
The polarity of the $\overline{PWRTE}$ bit	PWRTE	$\overline{PWRTE}$	$\overline{PWRTE}$	$\overline{PWRTE}$
Recommended value of REXT for RC oscillator circuits	REXT = 3kΩ - 100kΩ	REXT = 5kΩ - 100kΩ	REXT = 5kΩ - 100kΩ	REXT = 3kΩ - 100kΩ
GIE bit unintentional enable	If an interrupt occurs while the Global Interrupt Enable (GIE) bit is being cleared, the GIE bit may unintentionally be re-enabled by the user's Interrupt Service Routine (the RETFIE instruction).	N/A	N/A	N/A
Packages	PDIP, SOIC	PDIP, SOIC	PDIP, SOIC	PDIP, SOIC, SSOP
Open Drain High Voltage (VOD)	14V	12V	12V	8.5V

APPENDIX C: MIGRATION FROM BASELINE TO MID-RANGE DEVICES

This section discusses how to migrate from a baseline device (i.e., PIC16C5X) to a mid-range device (i.e., PIC16CXXX).

The following is the list of feature improvements over the PIC16C5X microcontroller family:

1. Instruction word length is increased to 14-bits. This allows larger page sizes, both in program memory (2K now as opposed to 512K before) and the register file (128 bytes now versus 32 bytes before).
2. A PC latch register (PCLATH) is added to handle program memory paging. PA2, PA1 and PA0 bits are removed from the STATUS register and placed in the OPTION register.
3. Data memory paging is redefined slightly. The STATUS register is modified.
4. Four new instructions have been added: RETURN, RETFIE, ADDLW, and SUBLW. Two instructions, TRIS and OPTION, are being phased out, although they are kept for compatibility with PIC16C5X.
5. OPTION and TRIS registers are made addressable.
6. Interrupt capability is added. Interrupt vector is at 0004h.
7. Stack size is increased to eight-deep.
8. RESET vector is changed to 0000h.
9. RESET of all registers is revisited. Five different RESET (and wake-up) types are recognized. Registers are reset differently.
10. Wake-up from SLEEP through interrupt is added.
11. Two separate timers, the Oscillator Start-up Timer (OST) and Power-up Timer (PWRT), are included for more reliable power-up. These timers are invoked selectively to avoid unnecessary delays on power-up and wake-up.
12. PORTB has weak pull-ups and interrupt-on-change features.
13. T0CKI pin is also a port pin (RA4/T0CKI).
14. FSR is a full 8-bit register.
15. "In system programming" is made possible. The user can program PIC16CXX devices using only five pins: VDD, VSS, VPP, RB6 (clock) and RB7 (data in/out).

To convert code written for PIC16C5X to PIC16F84A, the user should take the following steps:

1. Remove any program memory page select operations (PA2, PA1, PA0 bits) for CALL, GOTO.
2. Revisit any computed jump operations (write to PC or add to PC, etc.) to make sure page bits are set properly under the new scheme.
3. Eliminate any data memory page switching. Redefine data variables for reallocation.
4. Verify all writes to STATUS, OPTION, and FSR registers since these have changed.
5. Change RESET vector to 0000h.

INDEX

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web (WWW) site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape® or Microsoft® Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available by using your favorite Internet browser to attach to:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

ftp://ftp.microchip.com

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

Systems Information and Upgrade Hot Line

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits.The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

013001

PIC16F84A PRODUCT IDENTIFICATION SYSTEM

To order or obtain information (e.g., on pricing or delivery) refer to the factory or the listed sales office.

PART NO.	-XX	X	/XX	XXX
Device	Frequency Range	Temperature Range	Package	Pattern

Device	PIC16F84A[1], PIC16F84AT[2] PIC16LF84A[1], PIC16LF84AT[2]
Frequency Range	04 = 4 MHz 20 = 20 MHz
Temperature Range	- = 0°C to +70°C I = -40°C to +85°C
Package	P = PDIP SO = SOIC (Gull Wing, 300 mil body) SS = SSOP
Pattern	QTP, SQTP, ROM Code (factory specified) or Special Requirements . Blank for OTP and Windowed devices.

Examples:

a) PIC16F84A -04/P 301 = Commercial temp., PDIP package, 4 MHz, normal VDD limits, QTP pattern #301.

b) PIC16LF84A - 04I/SO = Industrial temp., SOIC package, 200 kHz, Extended VDD limits.

c) PIC16F84A - 20I/P = Industrial temp., PDIP package, 20 MHz, normal VDD limits.

Note 1: F = Standard VDD range
LF = Extended VDD range
2: T = in tape and reel - SOIC and SSOP packages only.

Sales and Support

Data Sheets
Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System
Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

Bibliografía y Software utilizado

Enlaces Consultados:

http://www.microchip.com/
http://www.ieespain.com/ieeproteus/isis.html
http://www.winpic800.com/
http://www.labcenter.com
http://www.ultraedit.com/loc/es/ultraedit_es.html
http://www.pic16f84a.org/
http://www.downarchive.com/software/office/366012-idm-ultracompare-pro-v80001010-x32x64eng-silent.html
http://www.pic16f84a.org/
http://www.superrobotica.com
http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010230
http://www.ariston.es/web/imgProductos/doc/DV164131.pdf
http://www.robotstorehk.com/srf04tech.pdf
http://tierra.aslab.upm.es/documents/PFC/PFC_EGilaberte.pdf
http://www.ondaradio.es/
http://es.rs-online.com/web/
http://es.wikipedia.org
http://www.powertip.com.tw/

Libros Consultados:

[1] Enrique Palacios, Fernando Remiro, Lucas J. López, *Microcontrolador PIC16F84, Desarrollo de Proyectos*, RA-MA, 2009

[2] Germán Tojeiro Calaza, *PROTEUS: Simulación de circuitos electrónicos y Microcontroladores a través de ejemplos*, Marcombo, 2009

Software Utilizado:

MPLAB
PROTEUS
KDIFF3
ULTRAEDIT
ULTRACOMPARE